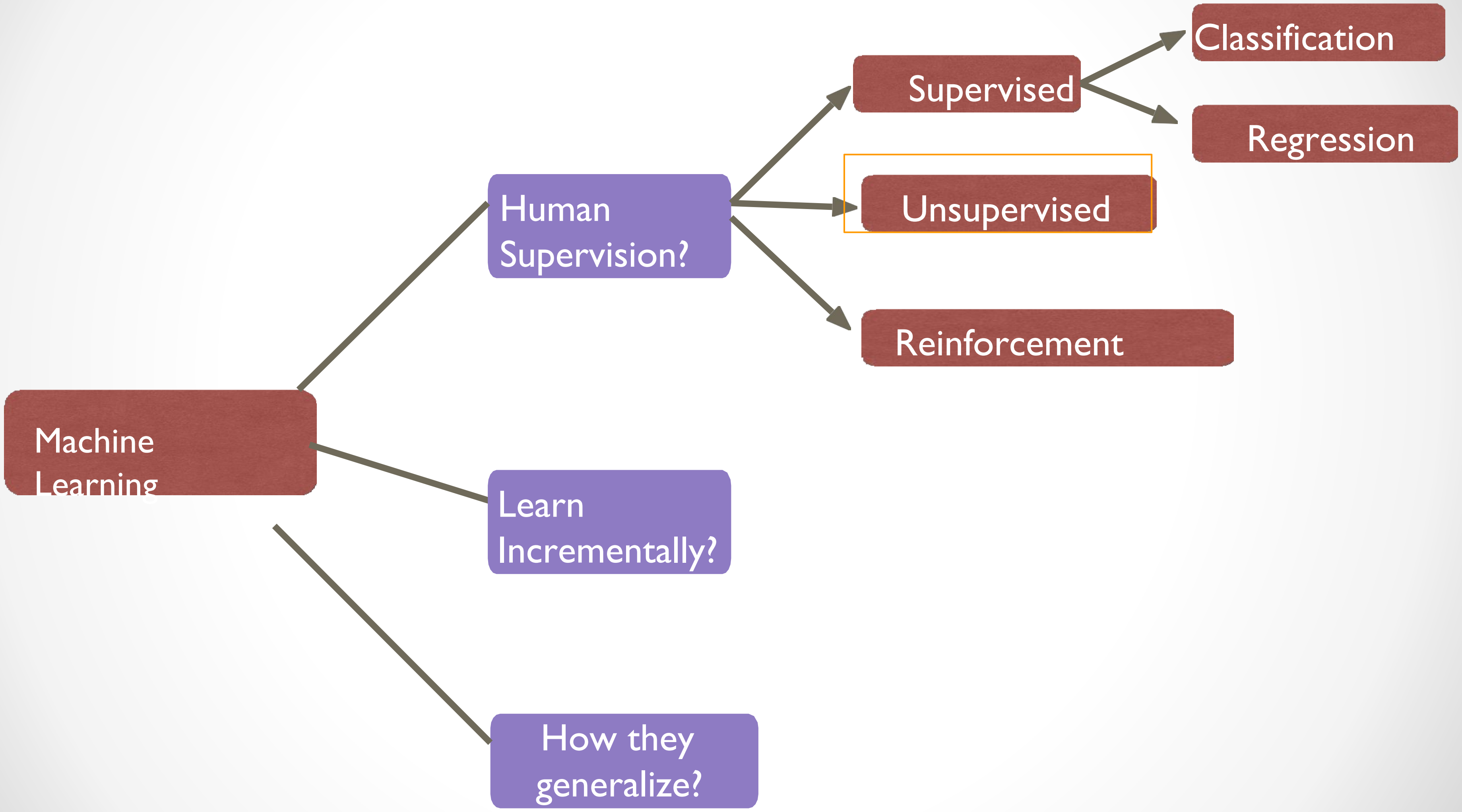
Unit -5

Chapter 9 Unsupervised Learning Techniques

What we will learn??

- Clustering
  - K-means
  - Limits of k-means
- Using Clustering for Image Segmentation
- Using Clustering for Pre-processing
- DBSCAN

Machine Learning - Types



Example of Unsupervised Machine Learning

- Let's, take the case of a baby and her family dog.

She knows and identifies this dog

Few weeks later a family friend brings along a dog and tries to play with the baby.

Baby has not seen this dog earlier.

But it recognizes many features (2 ears, eyes, walking on 4 legs) are like her pet dog.

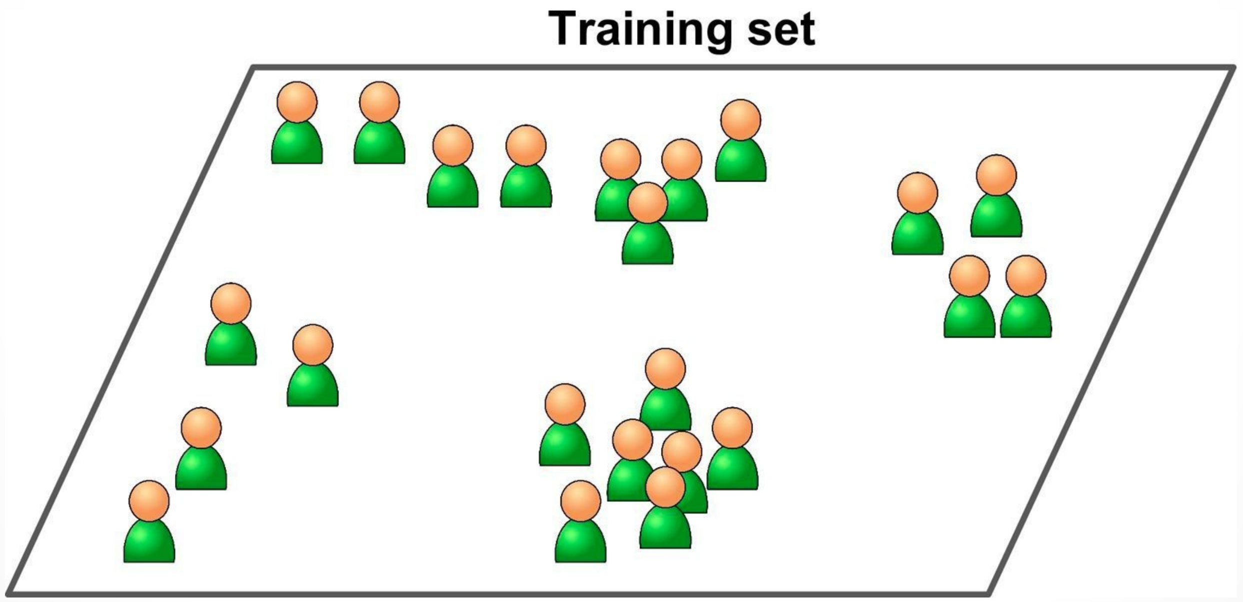
She identifies the new animal as a dog.

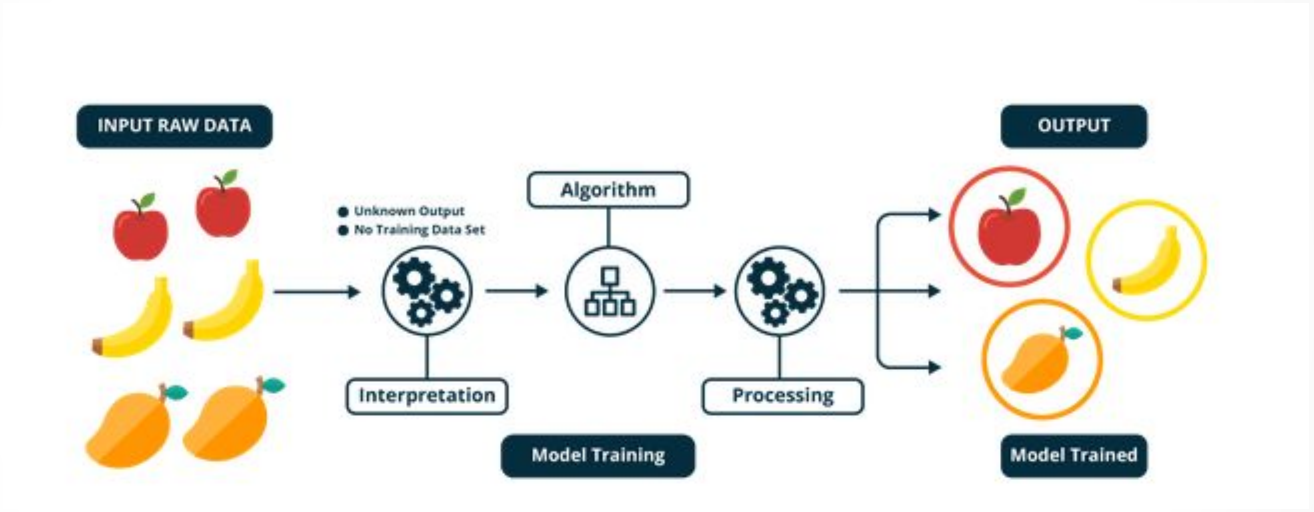
This is unsupervised learning, where you are not taught but you learn from the data (in this case data about a dog.)

Had this been supervised learning, the family friend would have told the baby that it's a dog.

Machine Learning - Unsupervised Learning

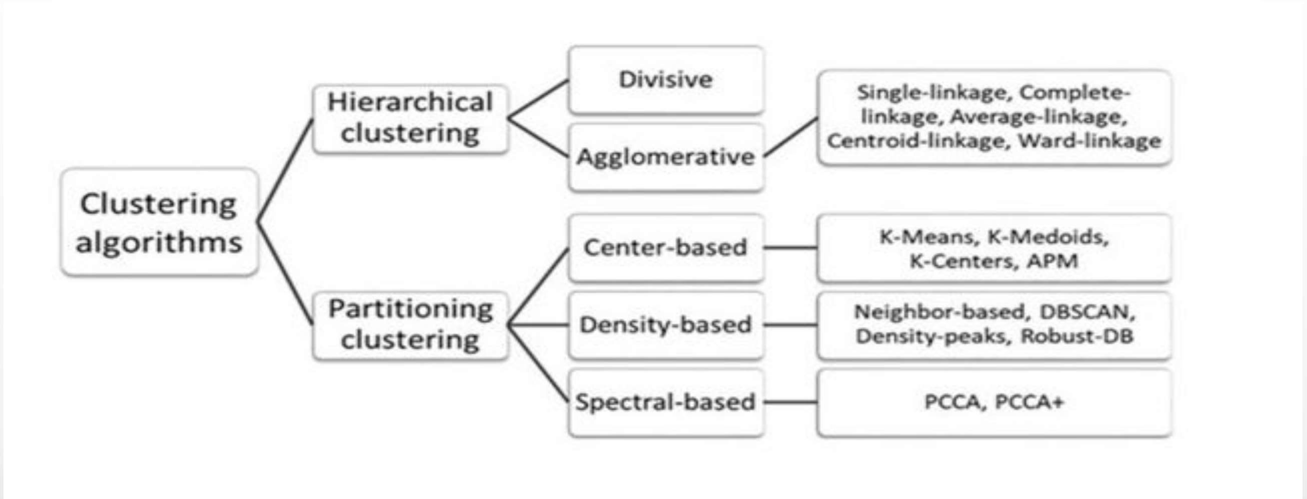
- The training data is unlabeled
- The system tries to learn without a teacher




INPUT RAW DATA
Unknown Output
No Training Data Set
Algorithm
Interpretation
Processing
Model Training
OUTPUT
Model Trained

Applications of Unsupervised Learning

- **The main applications of unsupervised learning includes:**
  - **Clustering**
    - For example, you can go to Walmart or a **supermarket** and see how different items are grouped and arranged there.
    - **e-commerce** websites like Amazon use clustering algorithms to implement a user-specific recommendation system.
    - **Libraries:** It is used in clustering different books on the basis of topic and information**.**


Clustering algorithms
Hierarchical clustering
Partitioning clustering
Divisive
Agglomerative
Single-linkage, Complete-linkage, Average-linkage, Centroid-linkage, Ward-linkage
Center-based
Density-based
Spectral-based
K-Means, K-Medoids, K-Centers, APM
Neighbor-based, DBSCAN, Density-peaks, Robust-DB
PCCA, PCCA+

- **Visualization** - Visualization is the process of creating diagrams, images, graphs, charts, etc., to communicate some information.

For example: Consider you are a football coach, and you have some data about your team's performance in a tournament. You may want to find all the statistics about the matches quickly.

You can feed the complex and unlabelled data to some visualization algorithm.

These algorithms will output a two-dimensional or three-dimensional representation of your data that can easily be plotted. So, by seeing the plotted graphs, you can easily get a lot of information.

- **Dimensionality reduction**

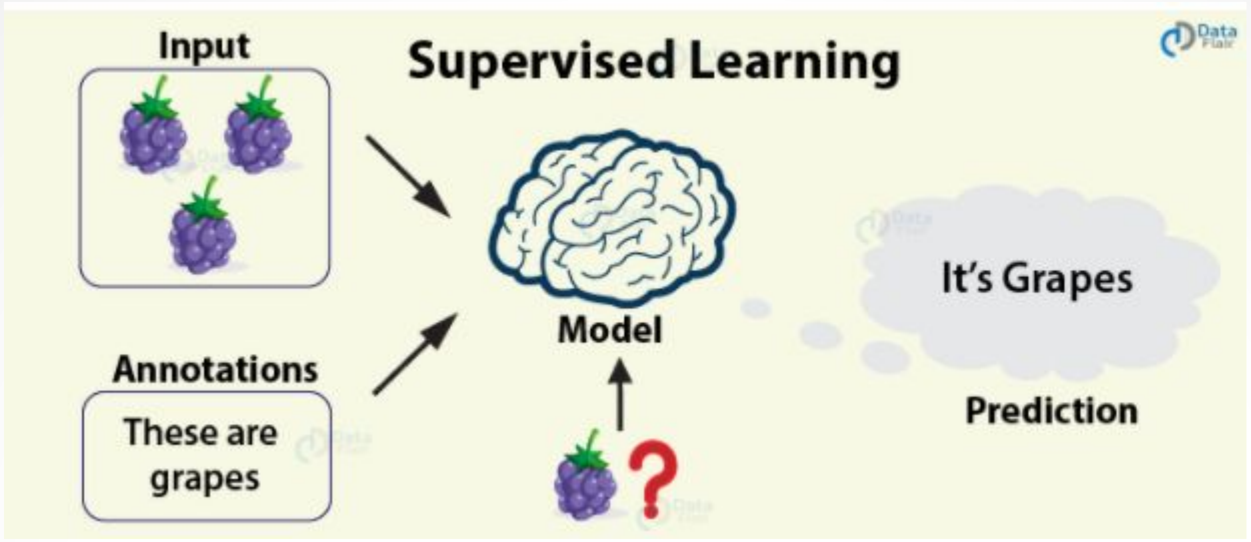
- **Finding association rules-** This is the process of finding associations between different parameters in the available data.
- The algorithm find some interesting relationships between attributes from large amounts of data .
- For example- when you visit website Amazon and buy some items, they will show you products similar to those as advertisements, even when you are not on their website.
- Amazon can find associations between different products and customers. They know that if they show a particular advertisement to a particular customer, chances are high that he will buy the product.
- Algorithms for association rule learning –

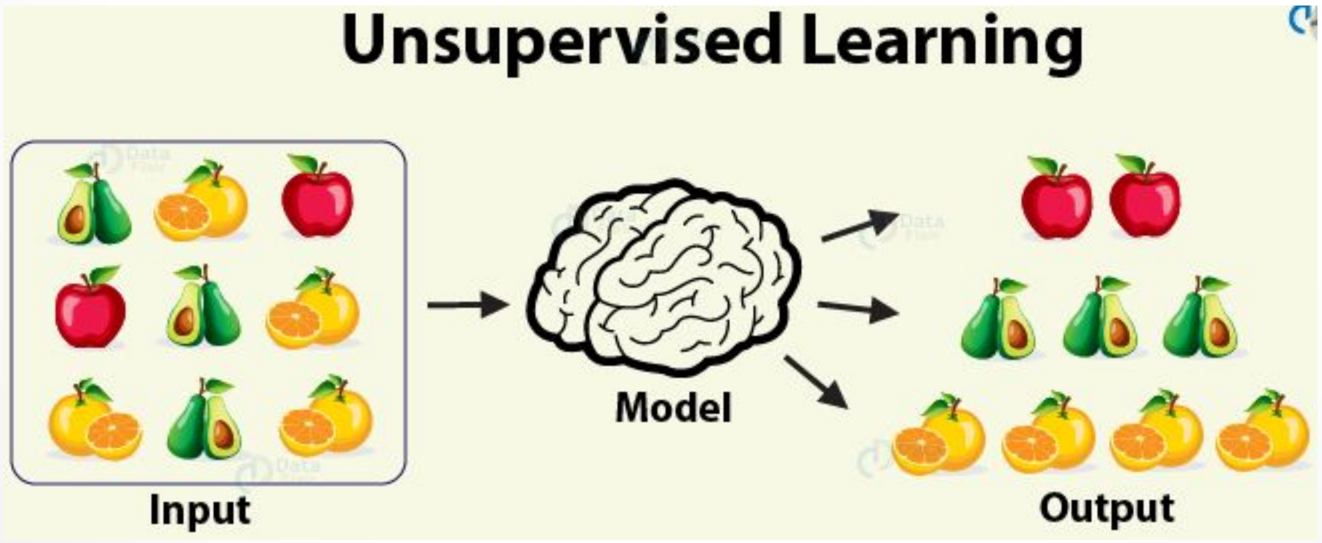
  1. Apriori
  2. Eclat

- **Anomaly detection-** Anomaly detection is the identification of rare items, events, or observations, which brings suspicions by differing significantly from the normal data.
- One important example of this is credit card fraud detection.

What is Unsupervised Learning Techniques

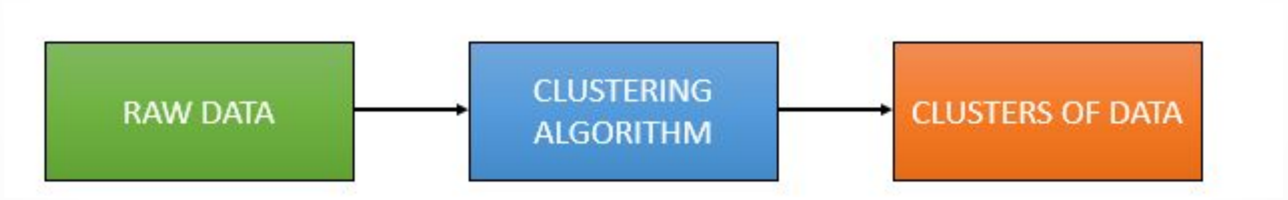
- **Unsupervised learning** is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision.
- Users do not need to supervise the model.
- Instead, it allows the model to work on its own to discover patterns and information that was previously undetected.
- **Unsupervised machine learning** helps you to finds all kind of unknown patterns in data.

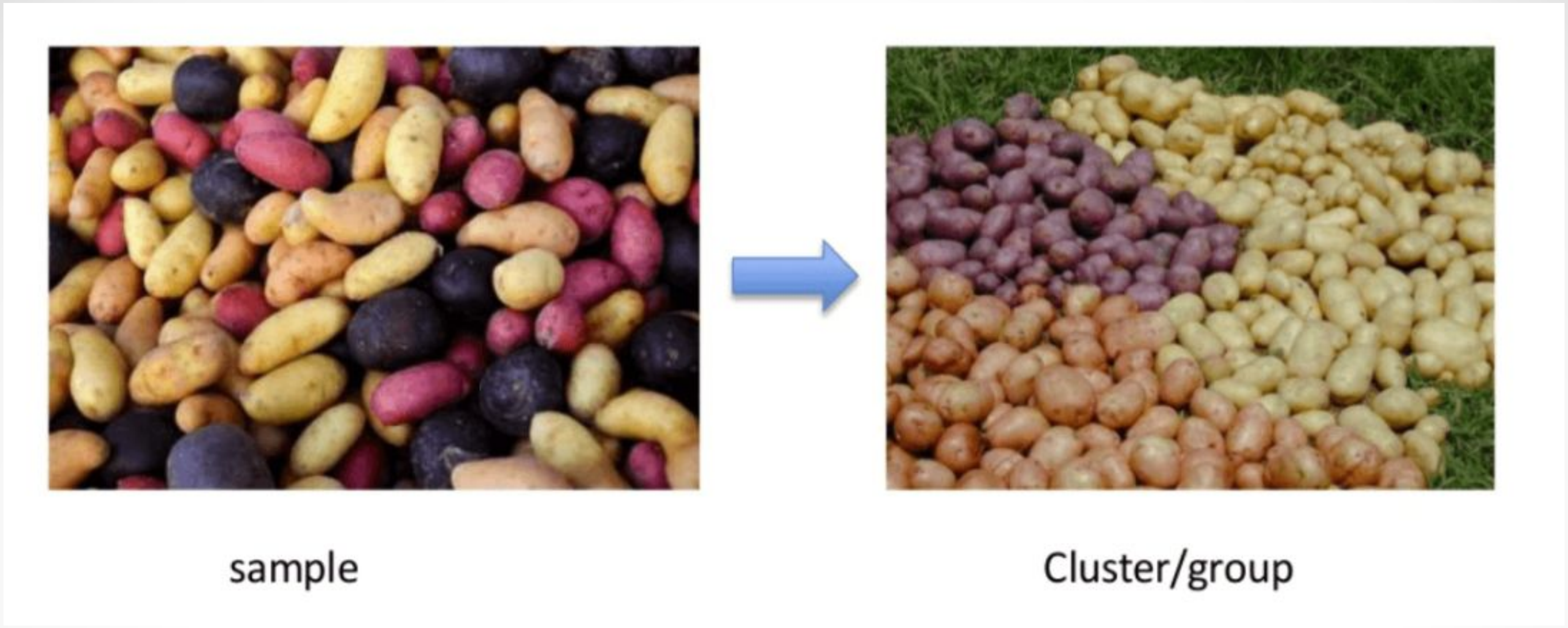

Input
Supervised Learning
Model
It's Grapes
Annotations
These are grapes
Prediction


Unsupervised Learning
Model
Input
Output

What is Clustering

- **Cluster analysis** or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar to each other than to those in other groups (clusters).





- Clustering mainly deals with finding a structure or pattern in a collection of uncategorized data.
- Clustering algorithms will process your data and find natural clusters(groups) if they exist in the data.

- Two popular clustering algorithms
  - K-Means
  - DBSCAN

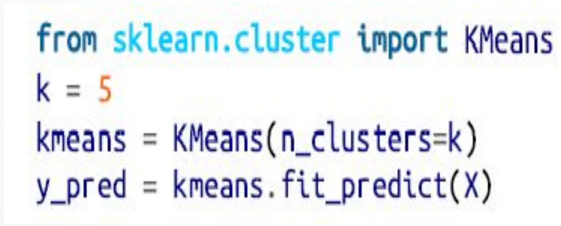
K-means

- The K-Means algorithm is capable of clustering unlabelled dataset very quickly and efficiently, in few iterations.
- It was proposed by Stuart Lloyd at Bell Labs in 1957 as a technique for pulse-code modulation.
- K-Means is sometimes referred to as Lloyd–Forgy.

Lloyd's(K-means) Algorithm

1. Pick k points at random from dataset => potential clusters and Initialize the centroids randomly
2. For every point in dataset:
   1. Compute distance to each cluster
   2. Assign to cluster with minimal distance
3. Compute new clusters' mean => updated clusters
4. Did clusters change significantly?
   1. Yes go to step 2 for refinement
   2. Otherwise stop

- Consider an unlabelled dataset and train a K-Means cluster on this dataset.
- Try to find each blob’s center and assign each instance to the closest blob.
- Looking at the data that k should be set to 5.
- Each instance was assigned to one of the five clusters



```python
from sklearn.cluster import KMeans
k = 5
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
```

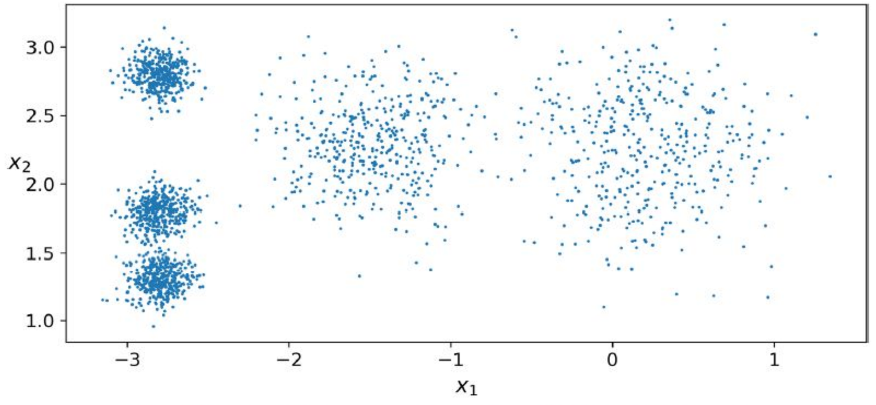
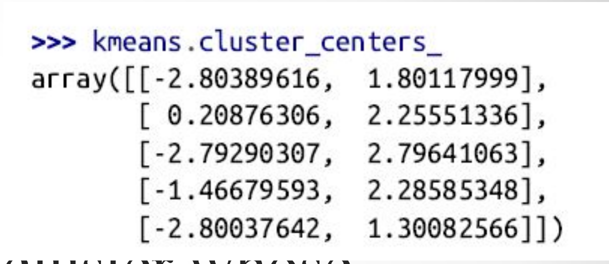


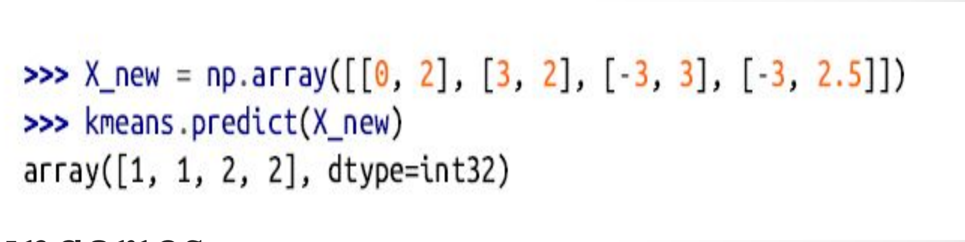
Figure 9-2. An unlabeled dataset composed of five blobs of instances

- Five centroids that the algorithm found



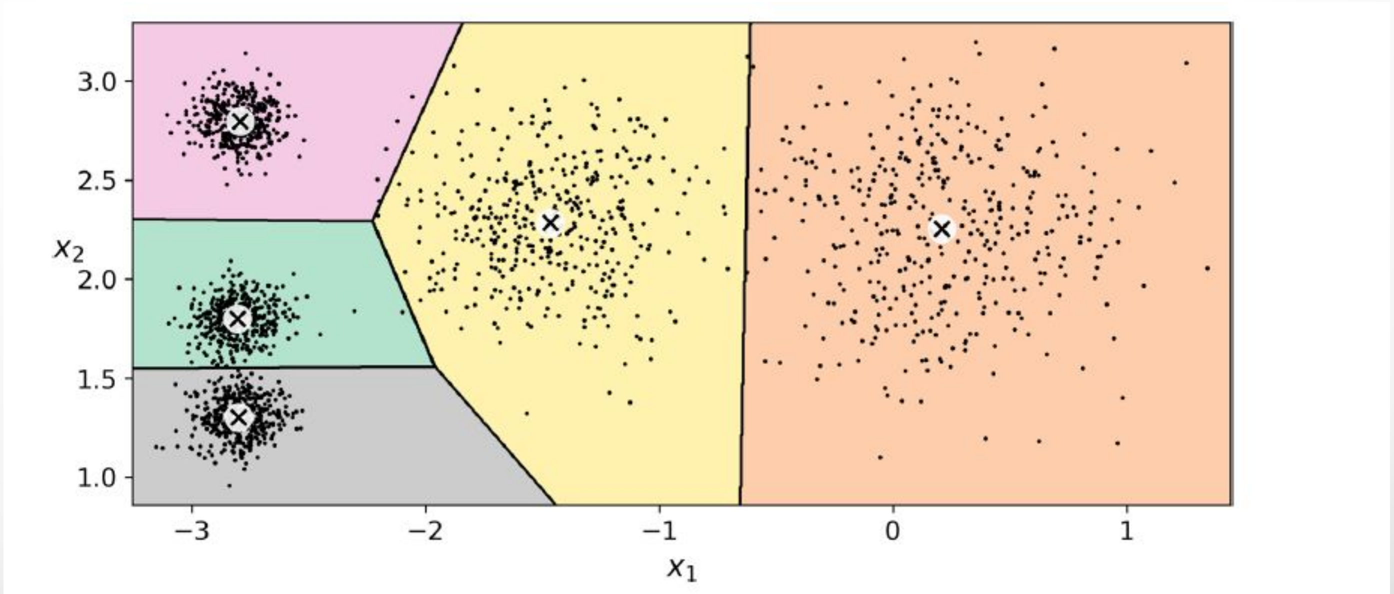
```
>>> kmeans.cluster_centers_
array([[-2.80389616,  1.80117999],
       [ 0.20876306,  2.25551336],
       [-2.79290307,  2.79641063],
       [-1.46679593,  2.28585348],
       [-2.80037642,  1.30082566]])
```

- You can easily assign new instances to the cluster whose centroid is closest.

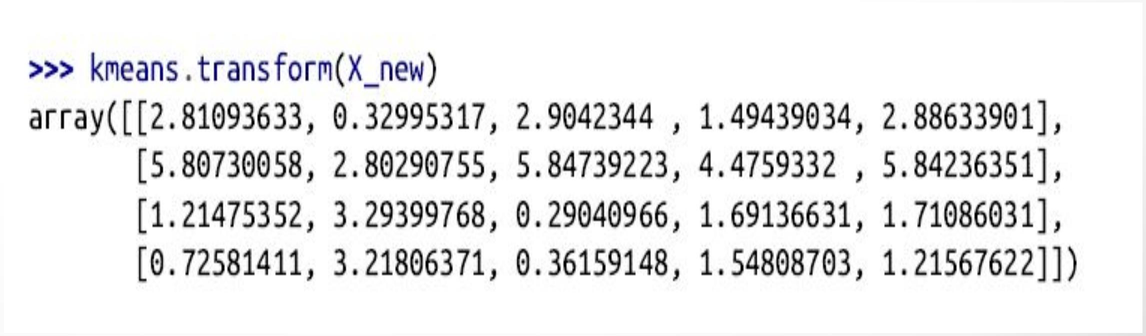


```
>>> X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]])
>>> kmeans.predict(X_new)
array([1, 1, 2, 2], dtype=int32)
```

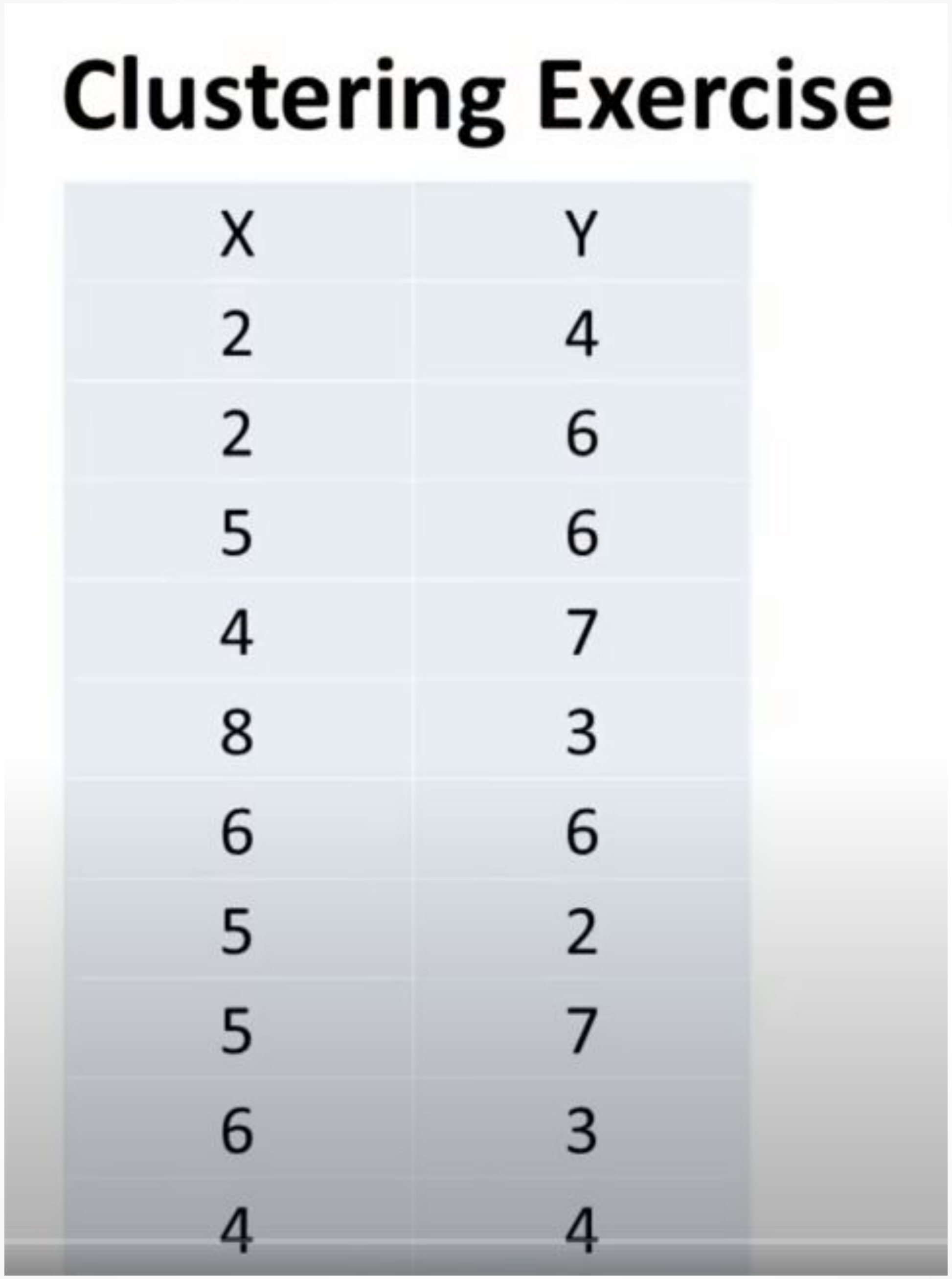
- plot the cluster’s decision boundaries



- The vast majority of the instances were clearly assigned to the appropriate cluster, but a few instances were probably mislabeled (especially near the boundary between the top-left cluster and the central cluster).
- Instead of assigning each instance to a single cluster, which is called **hard clustering**.
- Give each instance a score per cluster, which is called **soft clustering.**
- The score can be the distance between the instance and the centroid.
- The KMeans class, the transform() method measures the distance from each instance to every centroid:
- In this example, the first instance in X_new is located at a distance of 2.81 from the first centroid, 0.33 from the second centroid, 2.90 from the third centroid, 1.49 from the fourth centroid, and 2.89 from the fifth centroid.

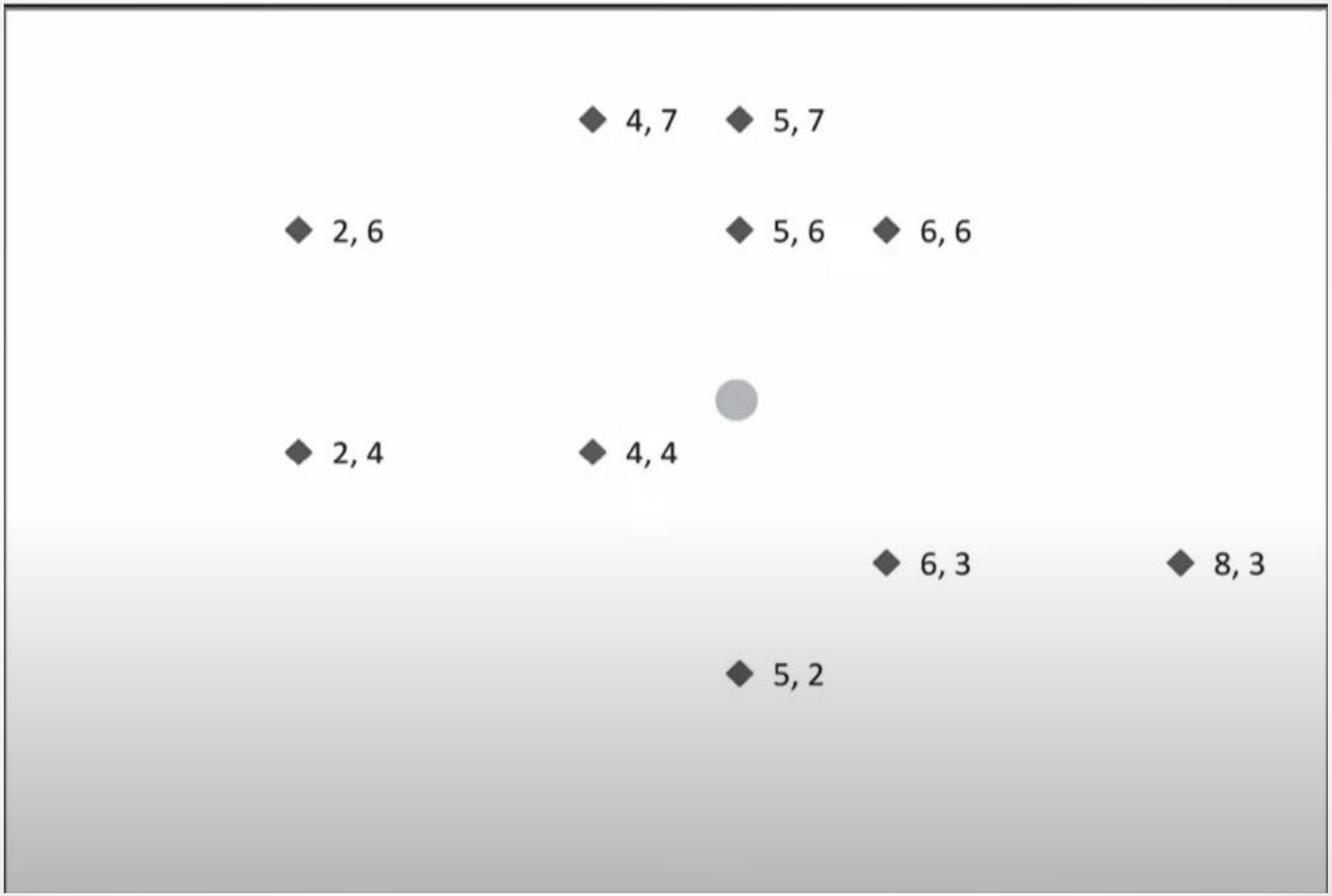


```
>>> kmeans.transform(X_new)
array([[2.81093633, 0.32995317, 2.9042344 , 1.49439034, 2.88633901],
       [5.80730058, 2.80290755, 5.84739223, 4.4759332 , 5.84236351],
       [1.21475352, 3.29399768, 0.29040966, 1.69136631, 1.71086031],
       [0.72581411, 3.21806371, 0.36159148, 1.54808703, 1.21567622]])
```

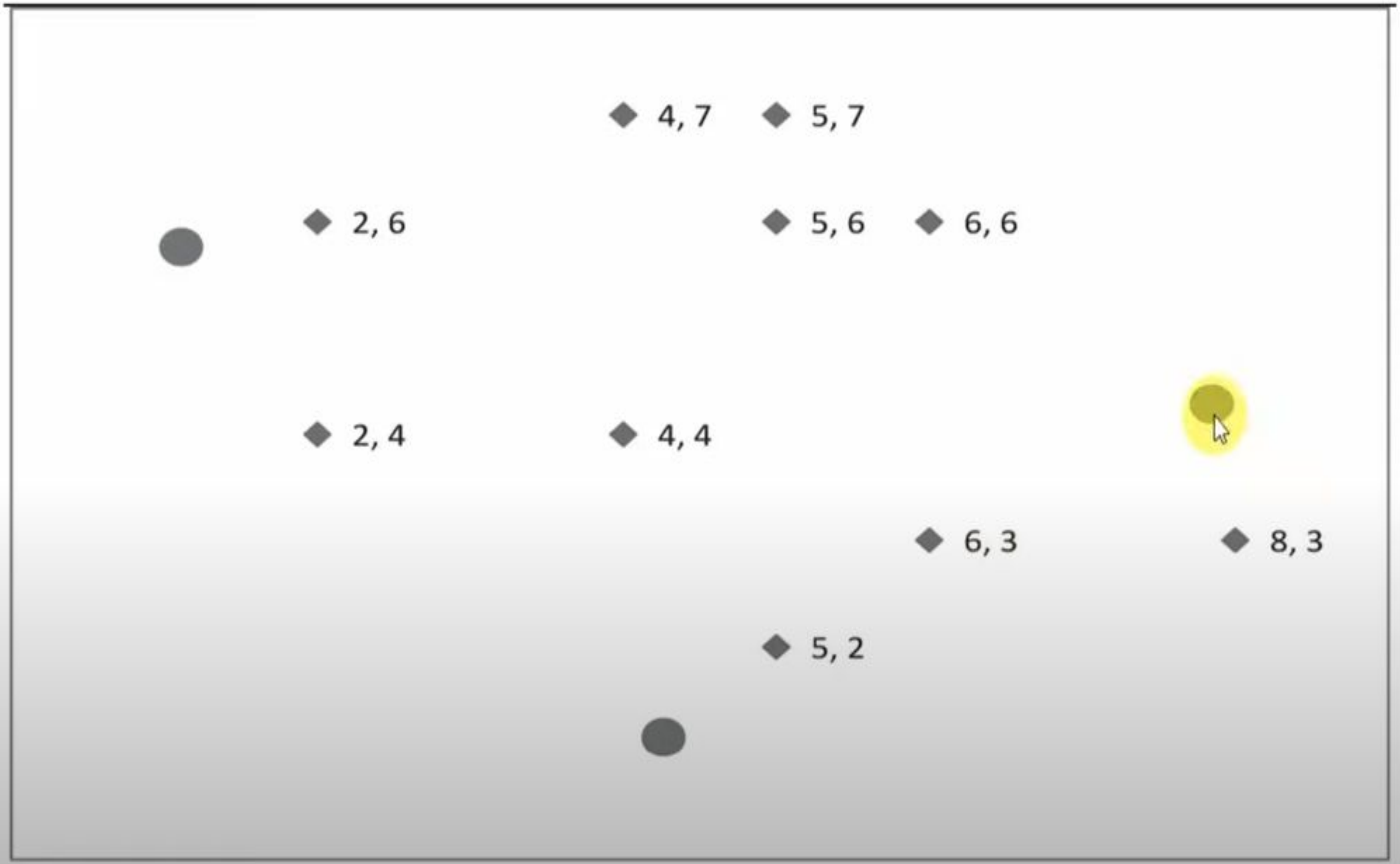


Clustering Exercise

X	Y
2	4
2	6
5	6
4	7
8	3
6	6
5	2
5	7
6	3
4	4


4, 7
5, 7
2, 6
5, 6
6, 6
2, 4
4, 4
6, 3
8, 3
5, 2

K-Means Algorithm for Clustering



Clustering Exercise

Iteration - 1

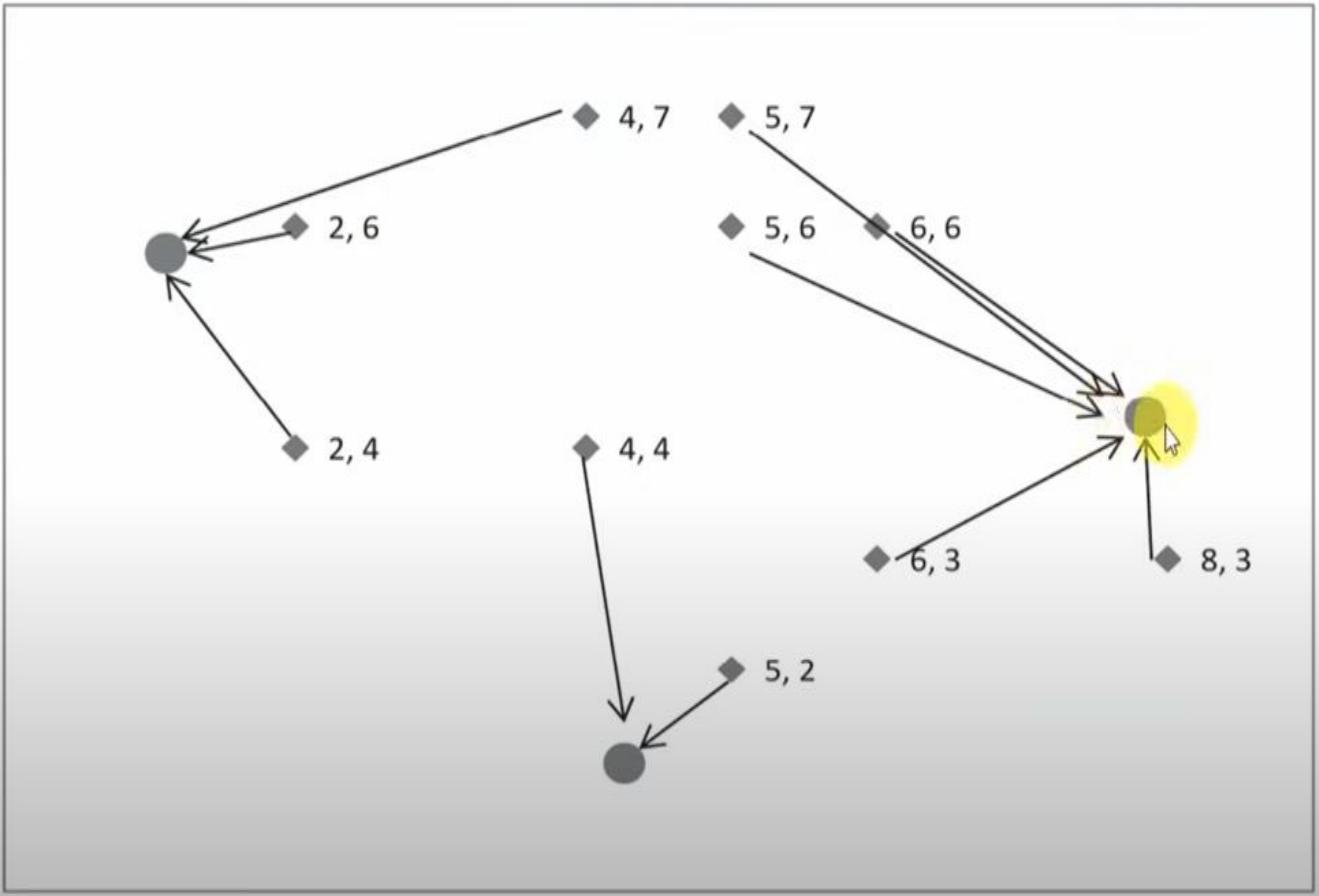
C1 - Seed Point1 – (1, 5)
C2 - Seed Point2 – (4, 1)
C3 - Seed Point3 – (8, 4)

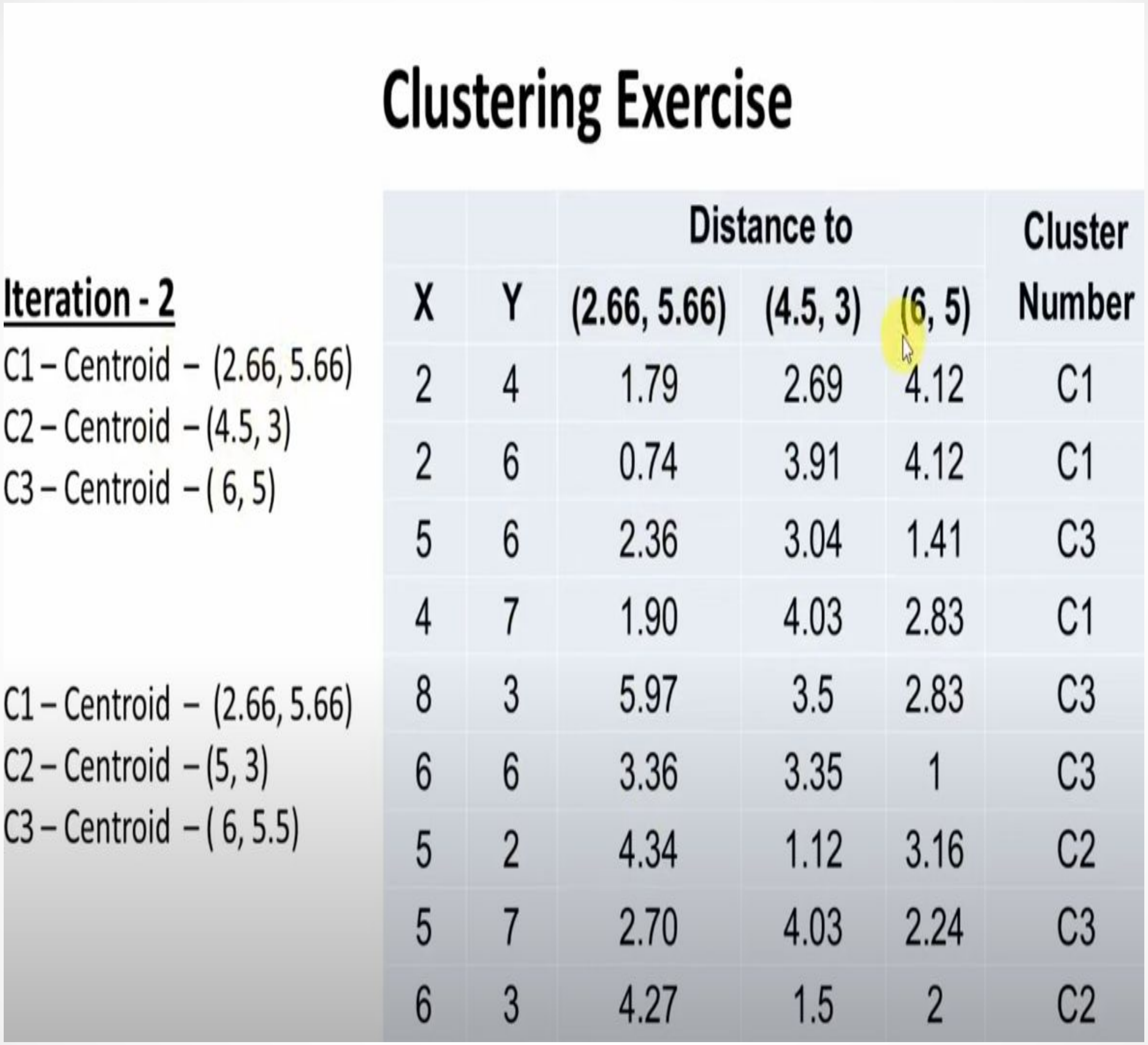
$$D = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$$

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (4.5, 3)
C3 – Centroid – (6, 5)

		Distance to			Cluster
X	Y	(1, 5)	(4, 1)	(8, 4)	Number
2	4	1.41	3.61	6.00	C1
2	6	1.41	5.39	6.32	C1
5	6	4.12	5.10	3.61	C3
4	7	3.61	6.00	5.00	C1
8	3	7.28	4.47	1.00	C3
6	6	5.10	5.39	2.83	C3
5	2	5.00	1.41	3.61	C2
5	7	4.47	6.08	4.24	C3
6	3	5.39	2.83	2.24	C3

K-Means Algorithm for Clustering





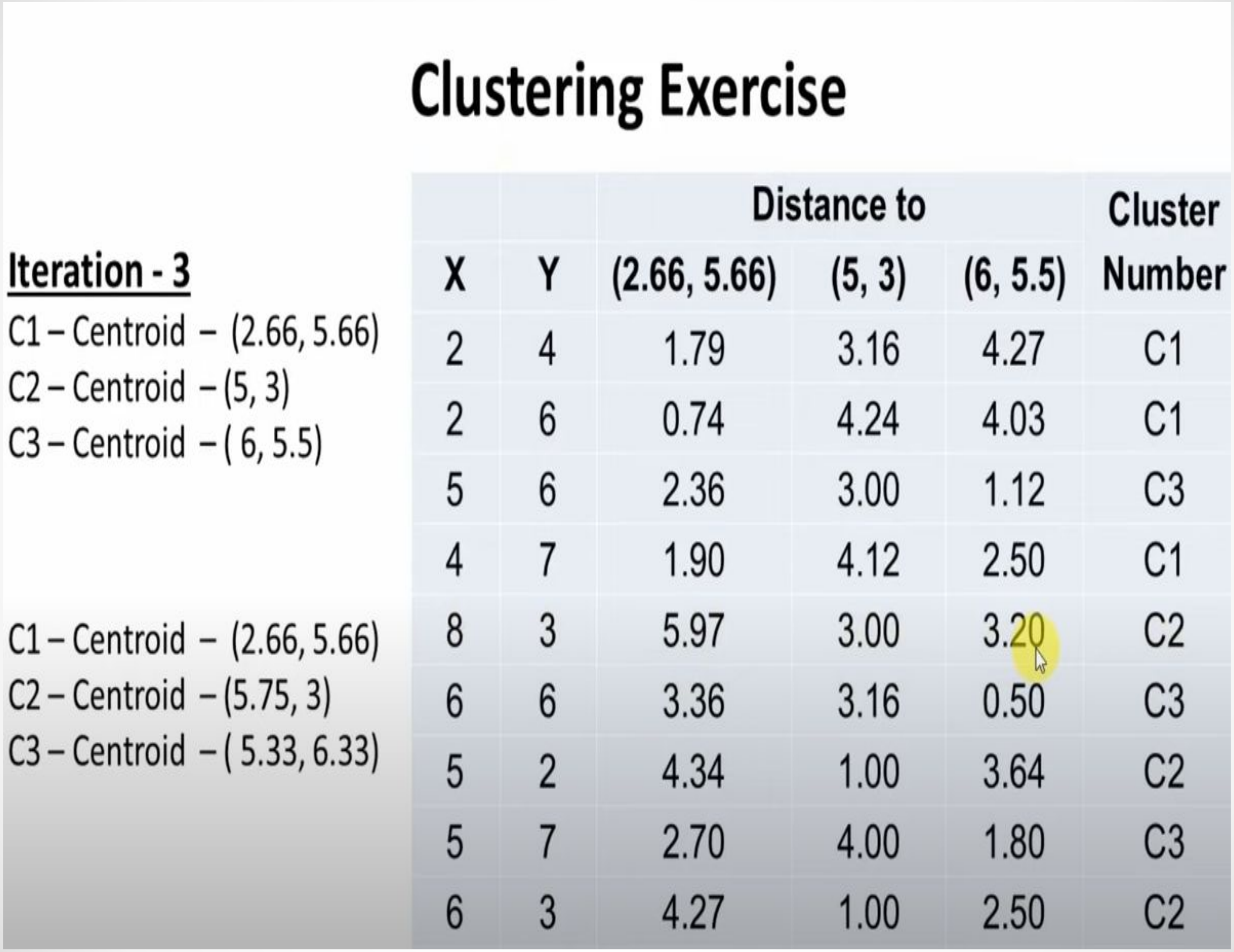
Clustering Exercise

Iteration - 2

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (4.5, 3)
C3 – Centroid – (6, 5)

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5, 3)
C3 – Centroid – (6, 5.5)

		Distance to			Cluster
X	**Y**	**(2.66, 5.66)**	**(4.5, 3)**	**(6, 5)**	**Number**
2	4	1.79	2.69	4.12	C1
2	6	0.74	3.91	4.12	C1
5	6	2.36	3.04	1.41	C3
4	7	1.90	4.03	2.83	C1
8	3	5.97	3.5	2.83	C3
6	6	3.36	3.35	1	C3
5	2	4.34	1.12	3.16	C2
5	7	2.70	4.03	2.24	C3
6	3	4.27	1.5	2	C2



Clustering Exercise

Iteration - 3

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5, 3)
C3 – Centroid – (6, 5.5)

		Distance to			Cluster
X	Y	(2.66, 5.66)	(5, 3)	(6, 5.5)	Number
2	4	1.79	3.16	4.27	C1
2	6	0.74	4.24	4.03	C1
5	6	2.36	3.00	1.12	C3
4	7	1.90	4.12	2.50	C1
8	3	5.97	3.00	3.20	C2
6	6	3.36	3.16	0.50	C3
5	2	4.34	1.00	3.64	C2
5	7	2.70	4.00	1.80	C3
6	3	4.27	1.00	2.50	C2

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5.75, 3)
C3 – Centroid – (5.33, 6.33)

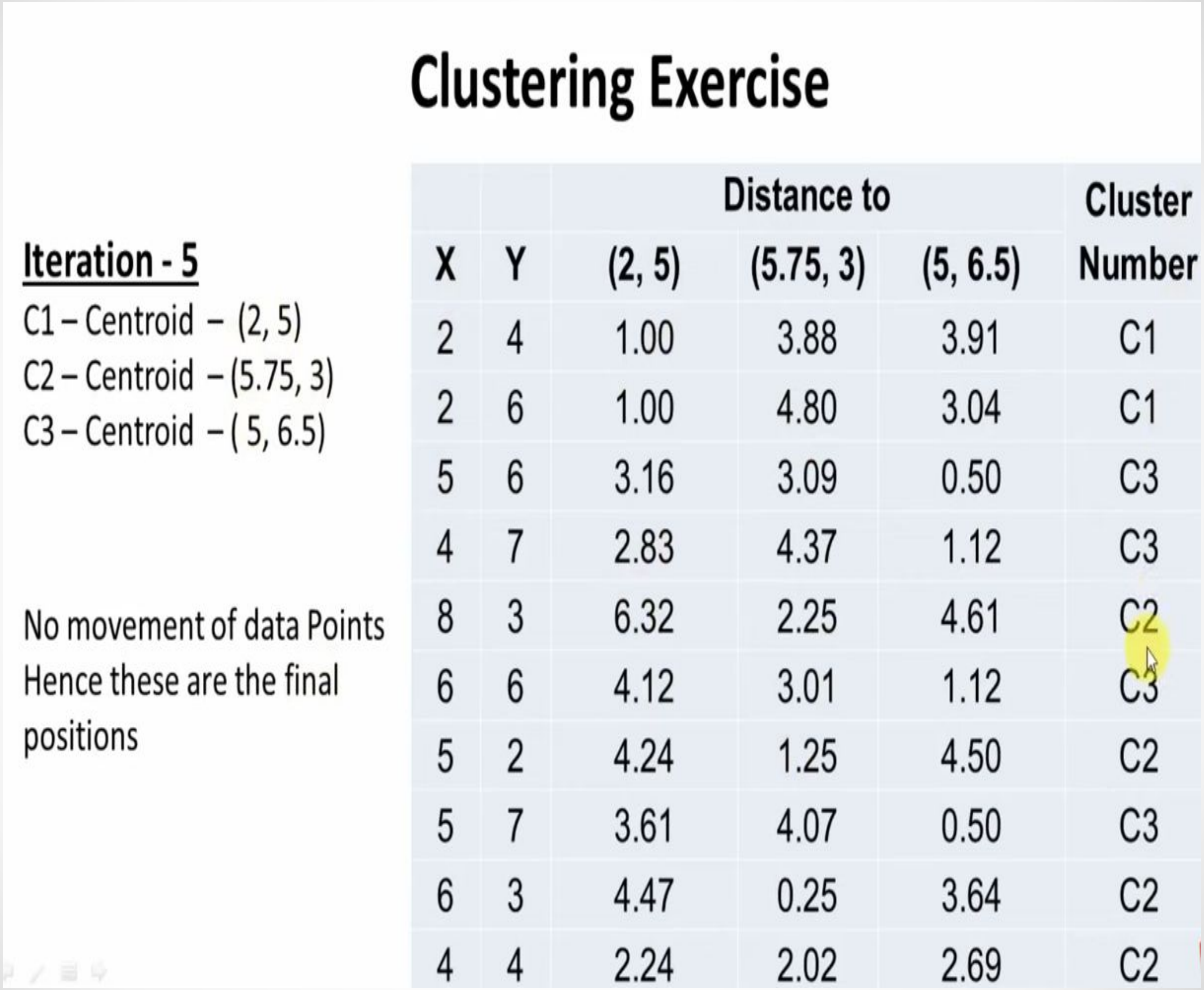
Clustering Exercise

<u>**Iteration - 4**</u>

C1 – Centroid – (2.66, 5.66)
C2 – Centroid – (5.75, 3)
C3 – Centroid – (5.33, 6.33)

		Distance to			Cluster
X	**Y**	**(2.66, 5.66)**	**(5.75, 3)**	**(5.33, 6.33)**	**Number**
2	4	1.79	3.88	4.06	C1
2	6	0.74	4.80	3.35	C1
5	6	2.36	3.09	0.47	C3
4	7	1.90	4.37	1.49	C3
8	3	5.97	2.25	4.27	C2
6	6	3.36	3.01	0.75	C3
5	2	4.34	1.25	4.34	C2
5	7	2.70	4.07	0.75	C3
6	3	4.27	0.25	3.40	C2
4	4	2.13	2.02	2.68	C2

C1 – Centroid – (2, 5)
C2 – Centroid – (5.75, 3)
C3 – Centroid – (5, 6.5)



Clustering Exercise

Iteration - 5

C1 – Centroid – (2, 5)

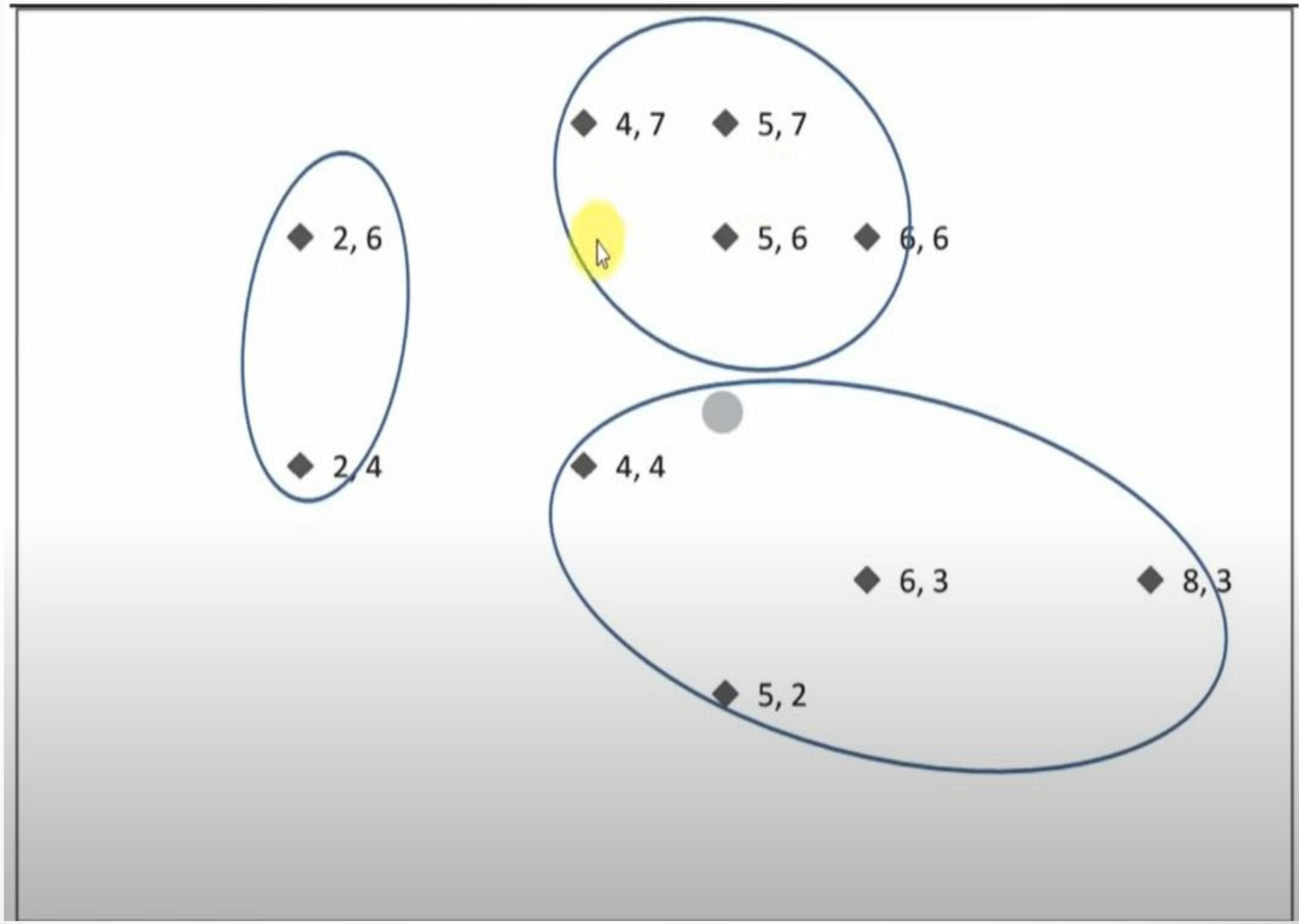
C2 – Centroid – (5.75, 3)

C3 – Centroid – (5, 6.5)

No movement of data Points Hence these are the final positions

		Distance to			Cluster
X	Y	(2, 5)	(5.75, 3)	(5, 6.5)	Number
2	4	1.00	3.88	3.91	C1
2	6	1.00	4.80	3.04	C1
5	6	3.16	3.09	0.50	C3
4	7	2.83	4.37	1.12	C3
8	3	6.32	2.25	4.61	C2
6	6	4.12	3.01	1.12	C3
5	2	4.24	1.25	4.50	C2
5	7	3.61	4.07	0.50	C3
6	3	4.47	0.25	3.64	C2
4	4	2.24	2.02	2.69	C2

Clustering Exercise



- K-means is one of the most popular clustering algorithms, mainly because of its good time performance.
- With the increasing size of the datasets being analysed, the computation time of K-means increases because of its constraint of needing the whole dataset in main memory.
- For this reason, several methods have been proposed to reduce the temporal and spatial cost of the algorithm.
  - **Mini batch K-means**
  - **Accelerated K-means**

Accelerated K-means

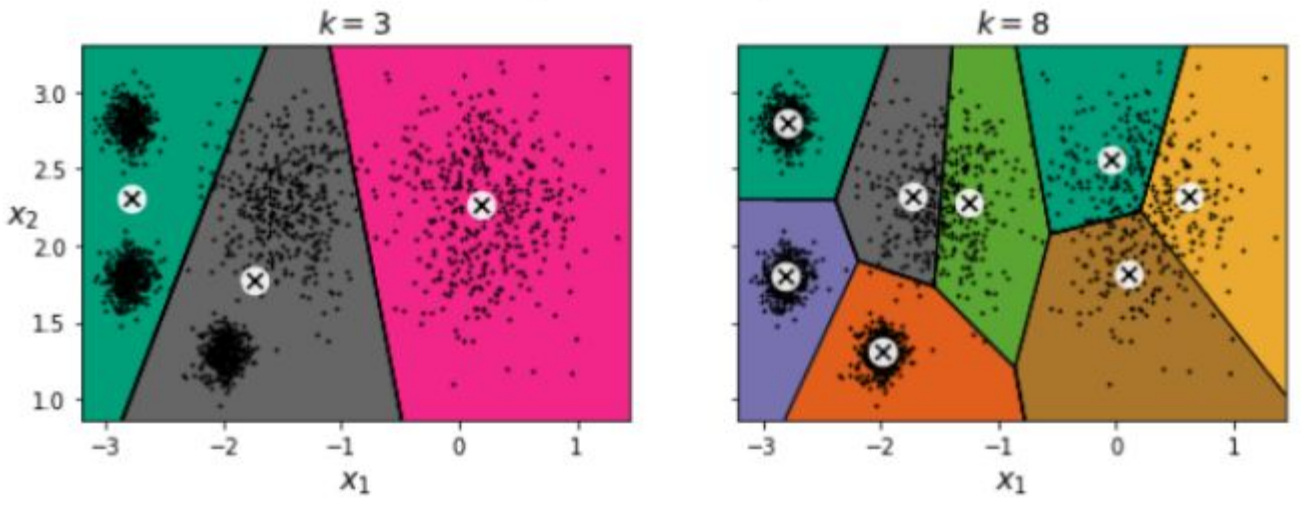
- Accelerated K-Means is the default for Sklearn.
- It considerably accelerates this algorithm by keeping track of the lower and upper bounds for the distances between instances and centroids.
- Accelerated k-means is a variant of the k-means algorithm. It uses some technology to accelerate the calculation.
- Like k-means, accelerated k-means also has two steps in each iteration: assigning each point to a cluster with closest distance to its centre; calculating new centre of each cluster. Moreover, the centre set and cluster partition set are exactly the same with the ordinary k-means after each iteration.
- The technologies used to accelerate iterations usually involve caches that store information between iterations. Therefore, it tends to use more memory than the ordinary k-means.

Mini-batch K-Means

- Instead of using the full dataset at each iteration, the algorithm is capable of using mini-batches, moving the centroid slightly at each iteration.
- This increases the speed of the algorithm by a factor of 3–4 typically.
- Especially important, it makes it possible to cluster huge datasets that do not fit in memory.
- One limitation is that its inertia is usually slightly worse, especially as clusters increase, however with many clusters the speed is much faster using mini-batch

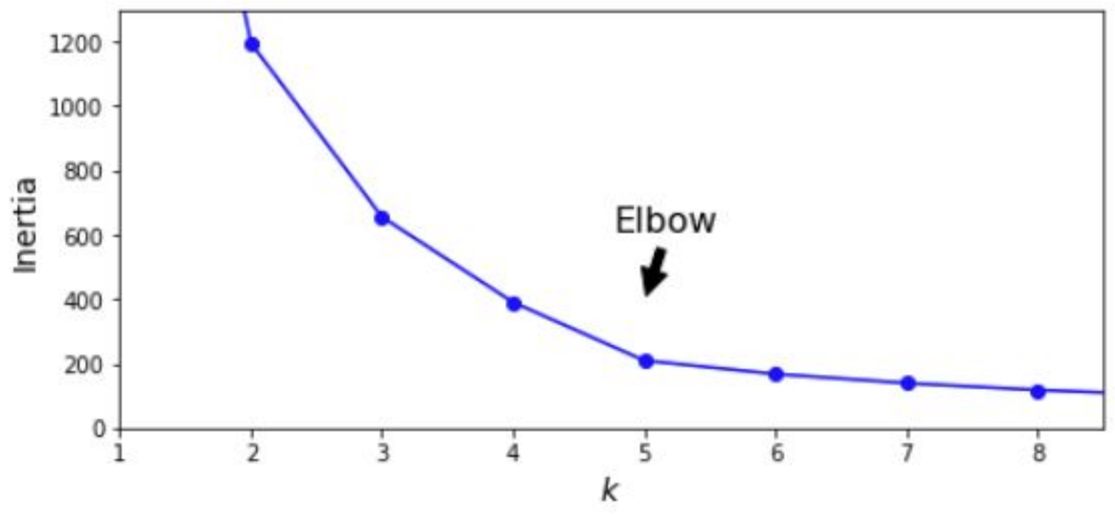
Finding the optimal number of clusters

In this dataset we can see that there are clearly 5 clusters we want to segment from each other. However that is not always the case, and often our data isn't as obviously segmented as this. Our result can be quite poor if we don't take precautions to figure out the optimal number of clusters:



Poor values for K

The first thought is to choose the k which minimises inertia, however we cannot do that as inertia will always decrease with a higher k. Indeed, the more clusters there are, the closer each instance will be to its closest centroid, and therefore the lower the inertia will be.
As we can see with a k=8, we are splitting clusters for no good reason.



Inertia as a function of k

Inertia drops very quickly up to 5, but then it decreases very slowly afterwards.
Any k lower than 5 and the gain would be dramatic, and any higher and we are not gaining much more information. This is a rather rough, subjective way of assigning k, however it generally works pretty well. When doing this it allows us to take context of a specific needs of a business problem too. However, in this case we could see that there were 5 clusters we wanted to segment. 4 clusters may be adequate but we should investigate the difference between k=4 and k=5

What is interia

- **Inertia** can be recognized as a measure of how internally coherent clusters are.
- **Inertia** is the sum of squared error for each **cluster**.
- Therefore the smaller the **inertia** the denser the **cluster**(closer together all the points are).
- The **KMeans** algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the **inertia** or within-cluster sum-of-squares(WCSS)

Limits of K-Means

The most important limitations of Simple k-means are:

- The user has to specify k (the number of clusters) in the beginning
- k-means can only handle numerical data
- k-means assumes that we deal with spherical clusters and that each cluster has roughly equal numbers of observations

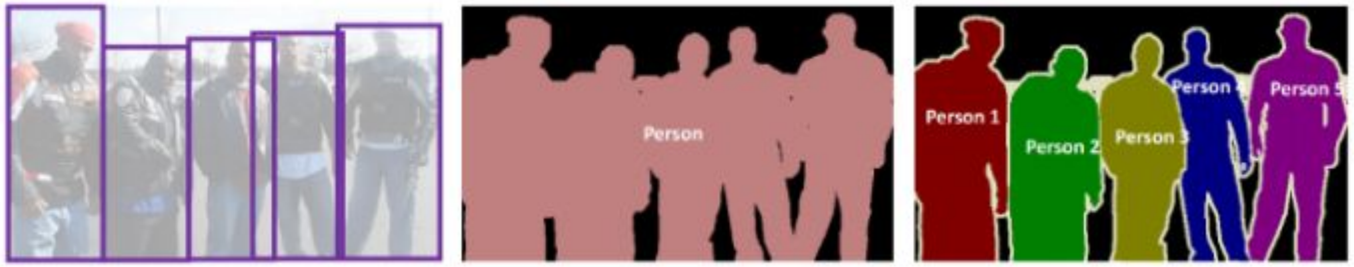
What is Image Segmentation?

- Imagine that you are going to cross the road, what you do before you cross the road?
- First, you see both sides of the road to determine the approaching vehicles and other environmental objects, then you do some amazing estimation of approaching speed and decide on when and how to cross the road. All these happens within a fraction of time, how amazing isn’t it.
- Our brain captures the Images of both sides of the road
- It detects the vehicles and other objects on the road == Object Detection
- Not only detect before that it determines the shape of every object it detects == Image Segmentation

Using Clustering for Image Segmentation

- Image segmentation is the task of partitioning an image into multiple segments.
- Image segmentation is broadly categorized into two main categories.
  - **Semantic Segmentation**
  - **Instance Segmentation**
- **Semantic segmentation** - all pixels that are part of the same object type get assigned to the same segment. A pixel-level classification is performed directly, we consider all those pixels belong to one class, so we represent them all by one color.
- **For example**- in a self-driving car's vision system, all pixels that are part of a pedestrian's image might be assigned to the "pedestrian" segment (there would be one segment containing all the pedestrians).

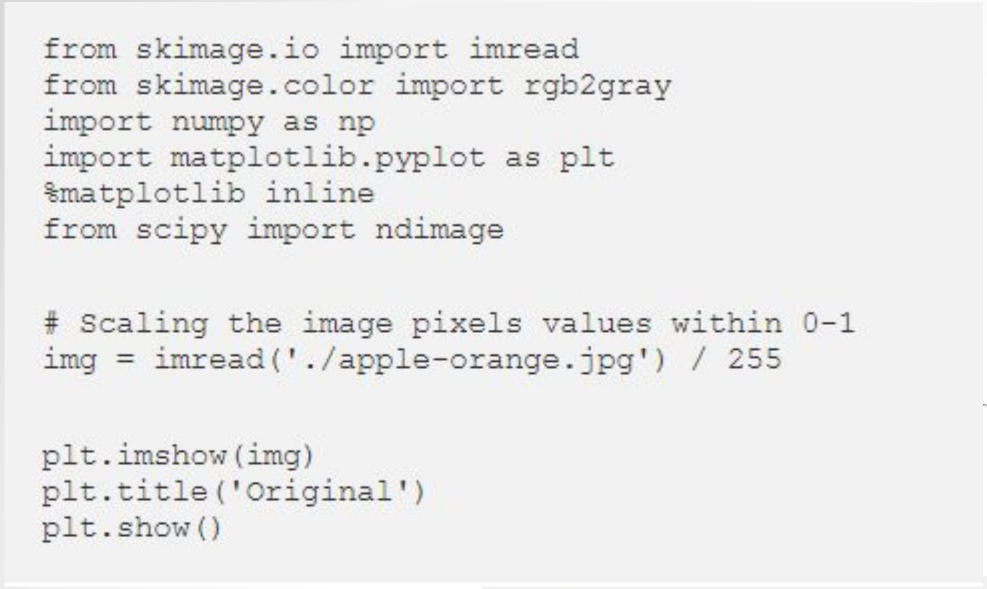
- **Instance segmentation -** All pixels that are part of the same individual object are assigned to the same segment.
- On the other hand in **instance segmentation**, approaches an additional object detection step is needed to obtain the individual instances of all classes in an image. Those pixels belong to the same class but we represent different instances of the same class with different colors.
- There would be a different segment for each pedestrian.
- We do it using *color segmentation -* assign pixels to the same segment if they have a similar color.
- **For example-** if you want to analyze satellite images to measure how much total forest area there is in a region, color segmentation may be just fine.



Objects Detected — Semantic Segments — Instance Segments PC: mc.ai

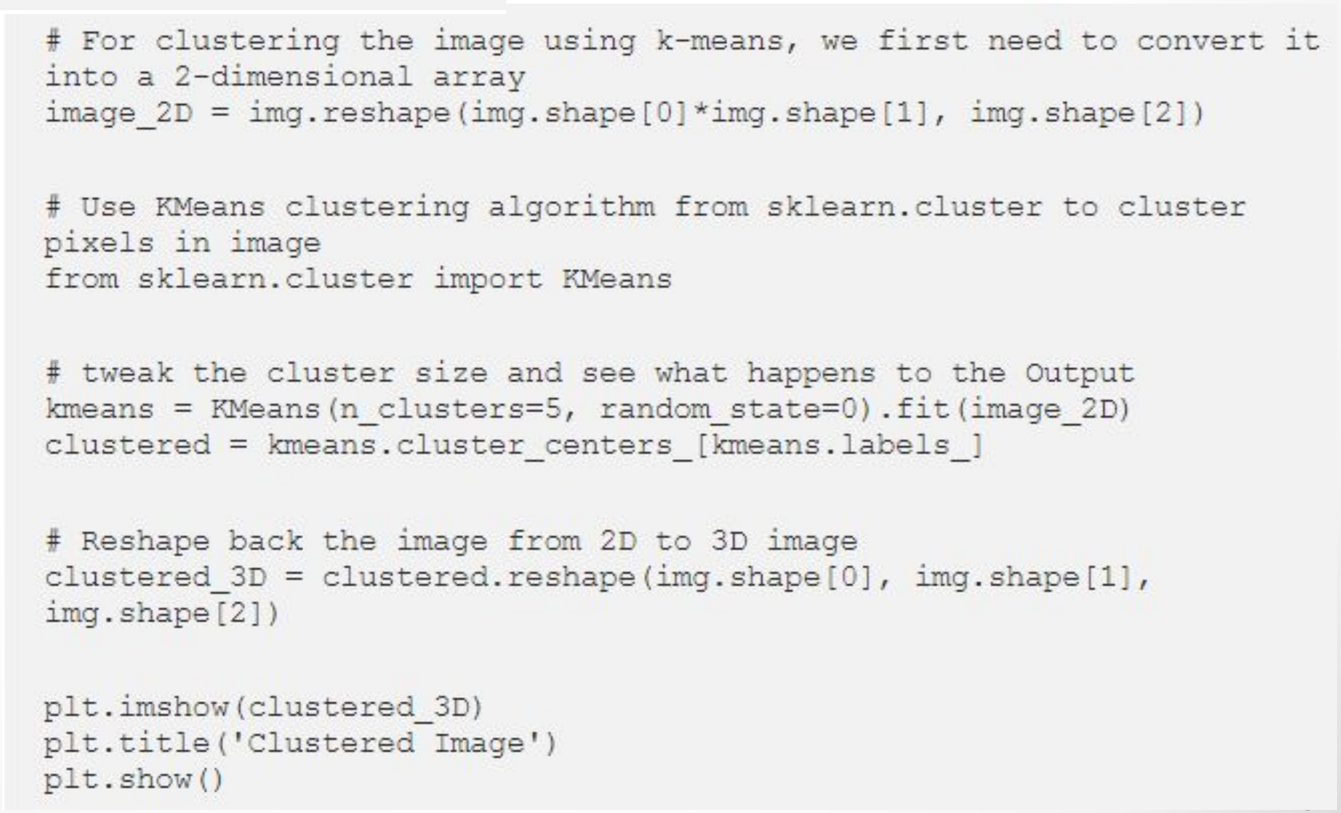
- In the first image, we can see that detected objects all are men.
- In **semantic segmentation**, we consider all those pixels belong to one class, so we represent them all by one color.
- On the other hand in **instance segmentation**, those pixels belong to the same class but we represent different instances of the same class with different colors.

- Clustering algorithms are used to group closer the data points that are more similar to each other, from other group data points.
- Now think of an image that holds apple and orange. Most of the pixel points in apple should be red/green, which is different from the pixel values of orange. If we can cluster these points we can distinguish each object from one another right. That’s how the cluster-based segmentation works. Let’s see some code samples now.



```
from skimage.io import imread
from skimage.color import rgb2gray
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from scipy import ndimage


# Scaling the image pixels values within 0-1
img = imread('./apple-orange.jpg') / 255


plt.imshow(img)
plt.title('Original')
plt.show()
```

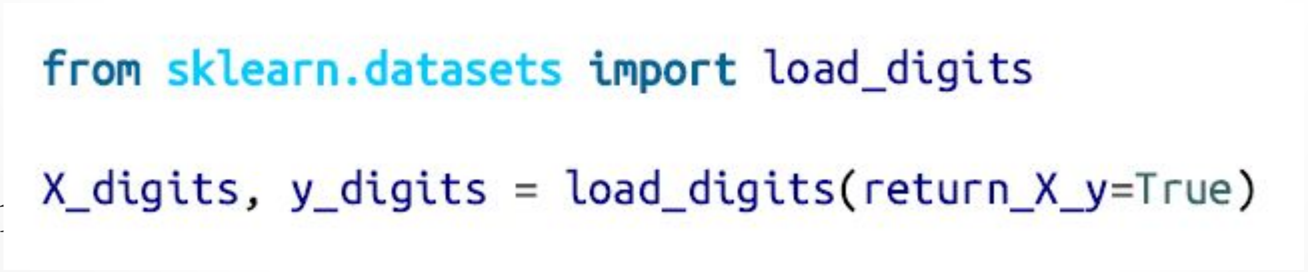
/cluster-based-image-segmentat



```
# For clustering the image using k-means, we first need to convert it
into a 2-dimensional array
image_2D = img.reshape(img.shape[0]*img.shape[1], img.shape[2])


# Use KMeans clustering algorithm from sklearn.cluster to cluster
pixels in image
from sklearn.cluster import KMeans


# tweak the cluster size and see what happens to the Output
kmeans = KMeans(n_clusters=5, random_state=0).fit(image_2D)
clustered = kmeans.cluster_centers_[kmeans.labels_]


# Reshape back the image from 2D to 3D image
clustered_3D = clustered.reshape(img.shape[0], img.shape[1],
img.shape[2])


plt.imshow(clustered_3D)
plt.title('Clustered Image')
plt.show()
```

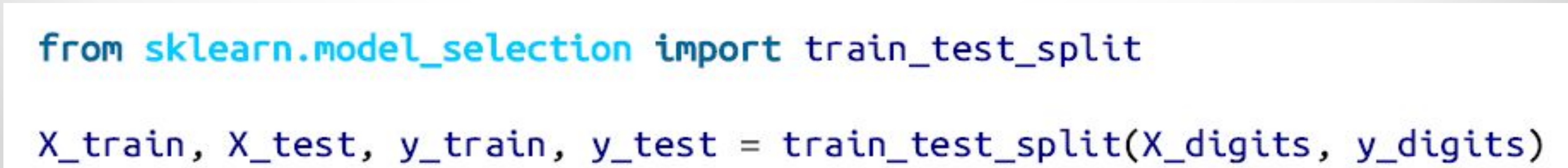
Using Clustering for Preprocessing

- Clustering can be an efficient approach as a pre-processing step before a supervised learning algorithm.
- MNIST-like dataset containing 1,797 grayscale 8 × 8 images representing the digits 0 to 9.
- First, load the dataset.



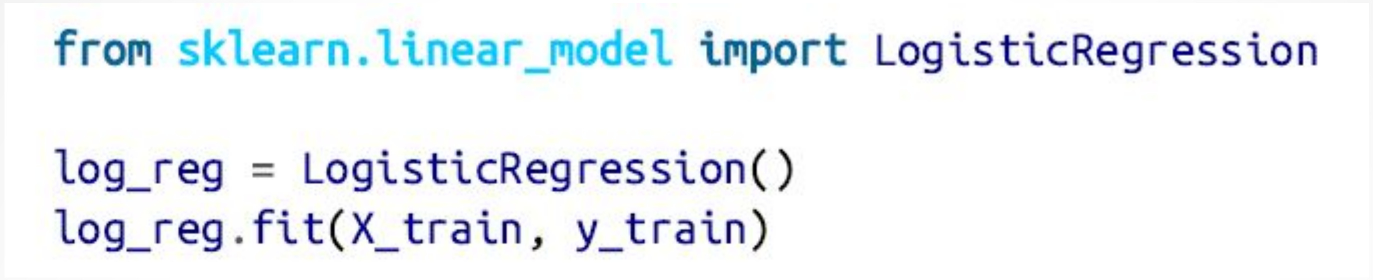
```python
from sklearn.datasets import load_digits

X_digits, y_digits = load_digits(return_X_y=True)
```

- split



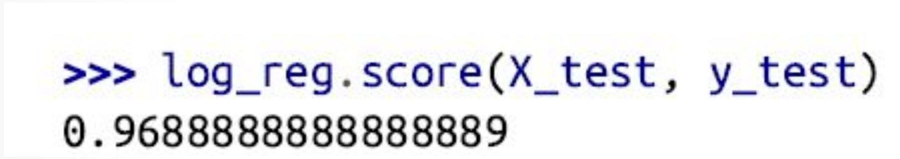
```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_digits, y_digits)
```

- fit a Logistic Regression model



```python
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```

- accuracy on the test set:



```
>>> log_reg.score(X_test, y_test)
0.9688888888888889
```

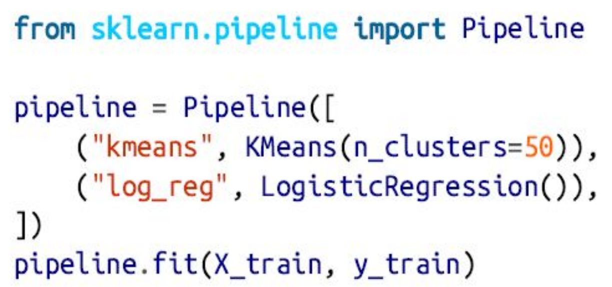
- Using K-Means as a pre-processing step, can do better

Step1 : first cluster the training set into 50 clusters.

Step 2: Replace the images with their distances to these 50 clusters.

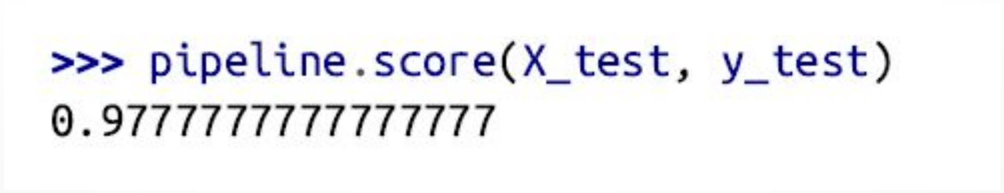
Step 3: apply a Logistic Regression model

Step 4: Evaluate this classification pipeline



```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ("kmeans", KMeans(n_clusters=50)),
    ("log_reg", LogisticRegression()),
])
pipeline.fit(X_train, y_train)
```

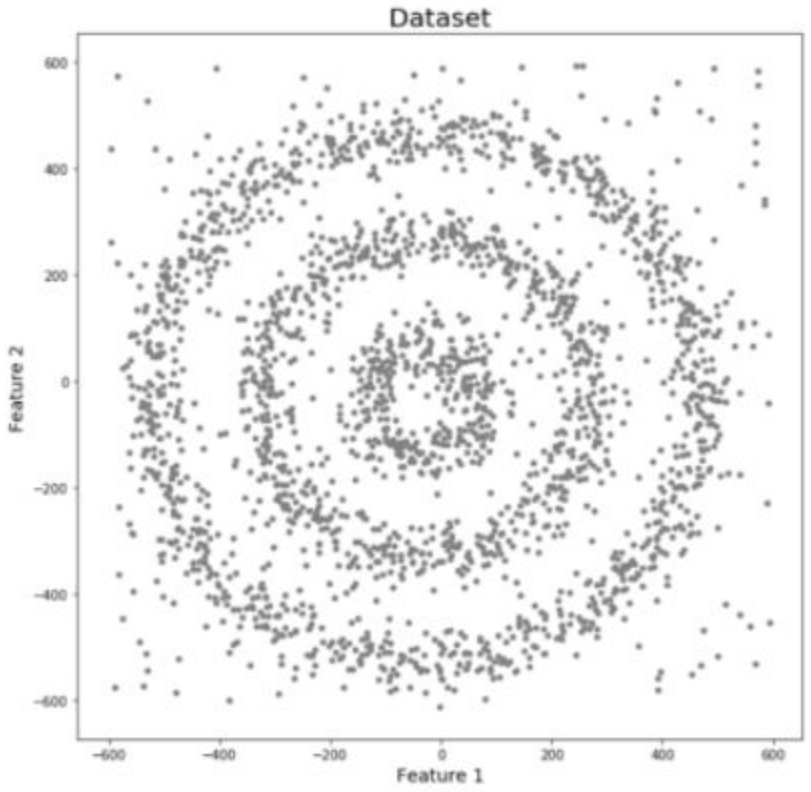
Reduced the error rate by almost 30%



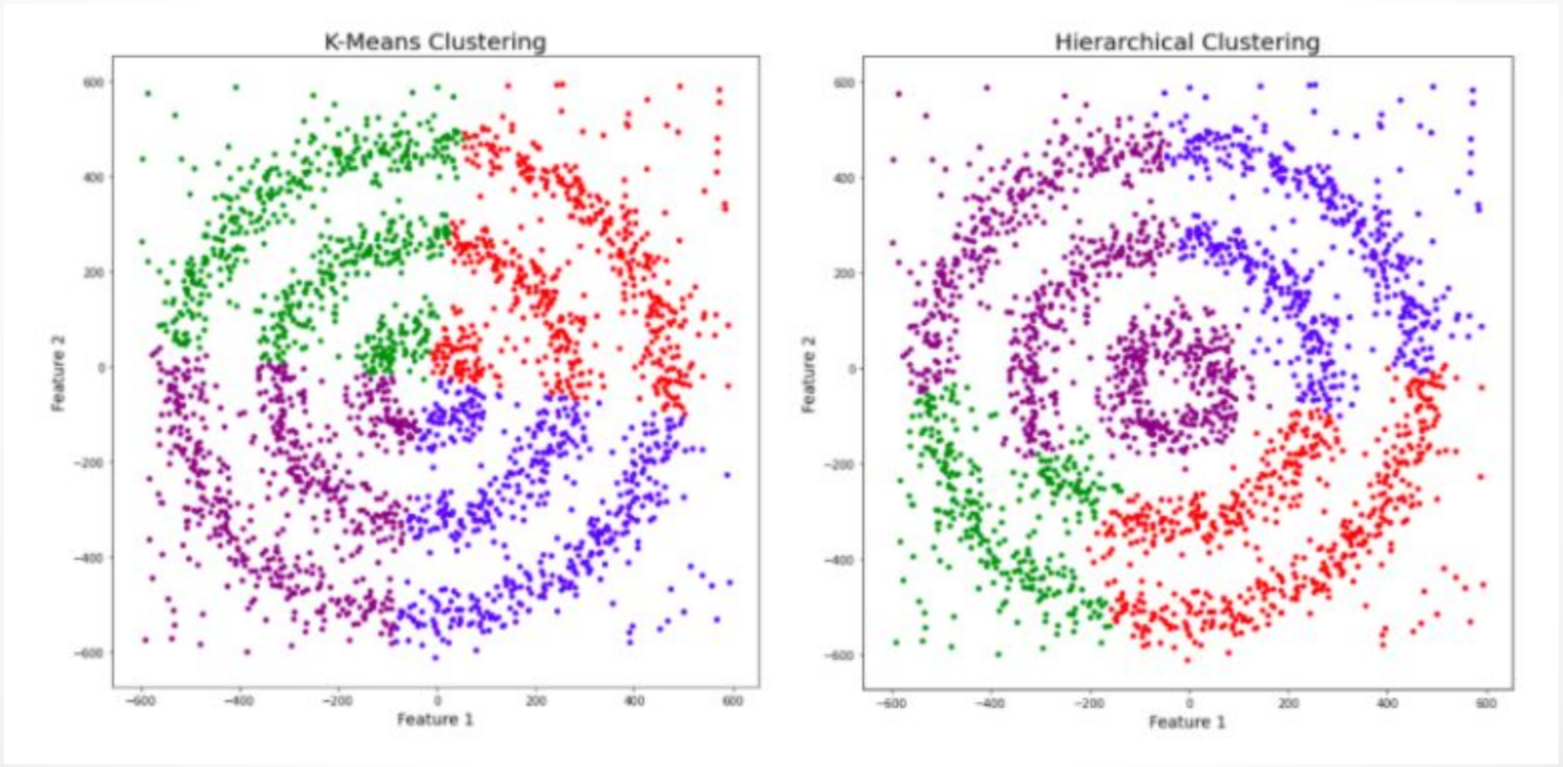
```
>>> pipeline.score(X_test, y_test)
0.9777777777777777
```

Why do we need DBSCAN Clustering?

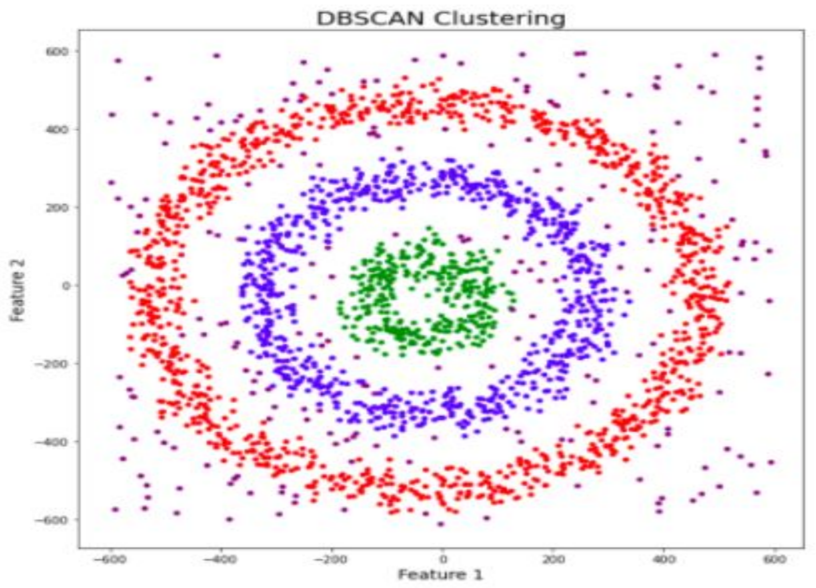
- Let's try to understand it with an example. Here we have data points densely present in the form of concentric circles:



- We can see three different dense clusters in the form of concentric circles with some noise here. Now, let’s run K-Means and Hierarchical clustering algorithms and see how they cluster these data points.



- We might be wondering why there are four colors in the graph?
- This data contains noise too and is represented by the purple color. Sadly, both of them failed to cluster the data points. Also, they were not able to properly detect the noise present in the dataset. Now, let’s take a look at the results from DBSCAN clustering.
- DBSCAN is not just able to cluster the data points correctly, but it also perfectly detects noise in the dataset.



What Exactly is DBSCAN Clustering?

- **DBSCAN** stands for **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise.
- *It was proposed by Martin Ester et al. in 1996. DBSCAN is a density-based clustering algorithm that works on the assumption that clusters are dense regions in space separated by regions of lower density.*
- It groups 'densely grouped' data points into a single cluster.
- **The most exciting feature of DBSCAN clustering is that it is robust to outliers**.
- It also does not require the number of clusters to be told beforehand, unlike K-Means, where we have to specify the number of centroids.

Essentials Terms

- **Epsilon(eps)**: The maximum distance a point can be from another point to be considered a neighbor.

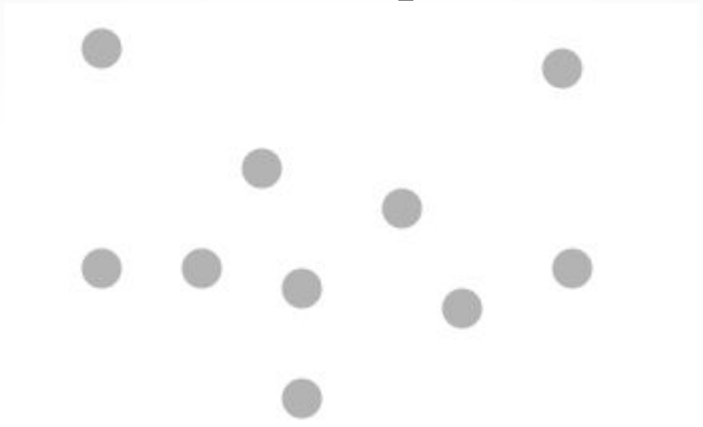
  or

  Epsilon is the radius of the circle to be created around each data point to check the density.

- **Min_Points(min_samples)**: The amount of points needed within the range of epsilon to be considered a cluster.
- The minimum number of data points required inside that circle for that data point to be classified as a **Core** point.
- **Metric**: The metric to use when calculating distance between instances in a feature array (i.e. euclidean distance).

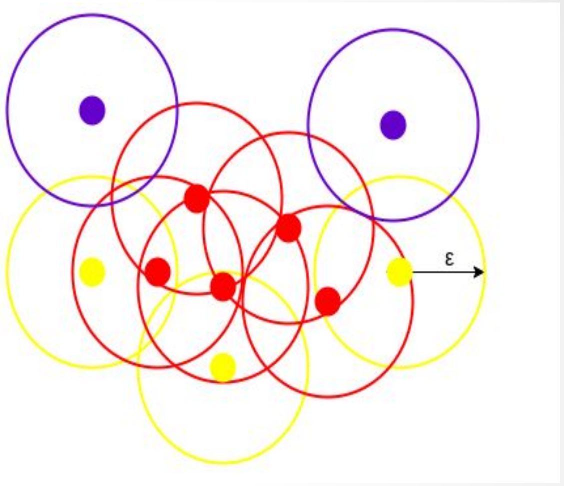
- The algorithm works by computing the distance between every point and all other points. We then place the points into one of three categories.
- **Core Points** — Points that have the Min Points required within epsilon (including itself). These points along with border points will form a cluster.
- **Border Points** — This is a point that has neighbors within epsilon but not enough neighbors to be a core point. These points make up the edge of the cluster.
- **Noise** — This is a point that does not have enough neighbors within epsilon to be part of a cluster (including itself).

- Let's understand it with the help of an example.



- Data points represented by grey color.
- Let's see how DBSCAN clusters these data points.
- DBSCAN creates a circle of *epsilon* radius around every data point and classifies them into **Core** point, **Border** point, and **Noise**.
- A data point is a **Core** point if the circle around it contains at least '*minPoints'* number of points. If the number of points is less than *minPoints*, then it is classified as **Border** Point, and if there are no other data points around any data point within *epsilon* radius, then it treated as **Noise**.

- The figure shows us a cluster created by DBCAN with *minPoints = 3*.
- Here, we draw a circle of equal radius *epsilon* around every data point. These two parameters help in creating spatial clusters.
- All the data points with at least 3 points in the circle including itself are considered as **Core** points represented by **red color.**
- All the data points with less than 3 but greater than 1 point in the circle including itself are considered as **Border** points. They are represented by **yellow color**.
- Finally, data points with no point other than itself present inside the circle are considered as **Noise** represented by the **purple color**.
- For locating data points in space, DBSCAN uses Euclidean distance.



DBSCAN Algorithm

- Label points as core, border and noise
- Eliminate noise points
- For every core point p that has not been assigned to a cluster
    - Create a new cluster with the point p and all the points that are density-connected to p.
- Assign border points to the cluster of the closest core point.

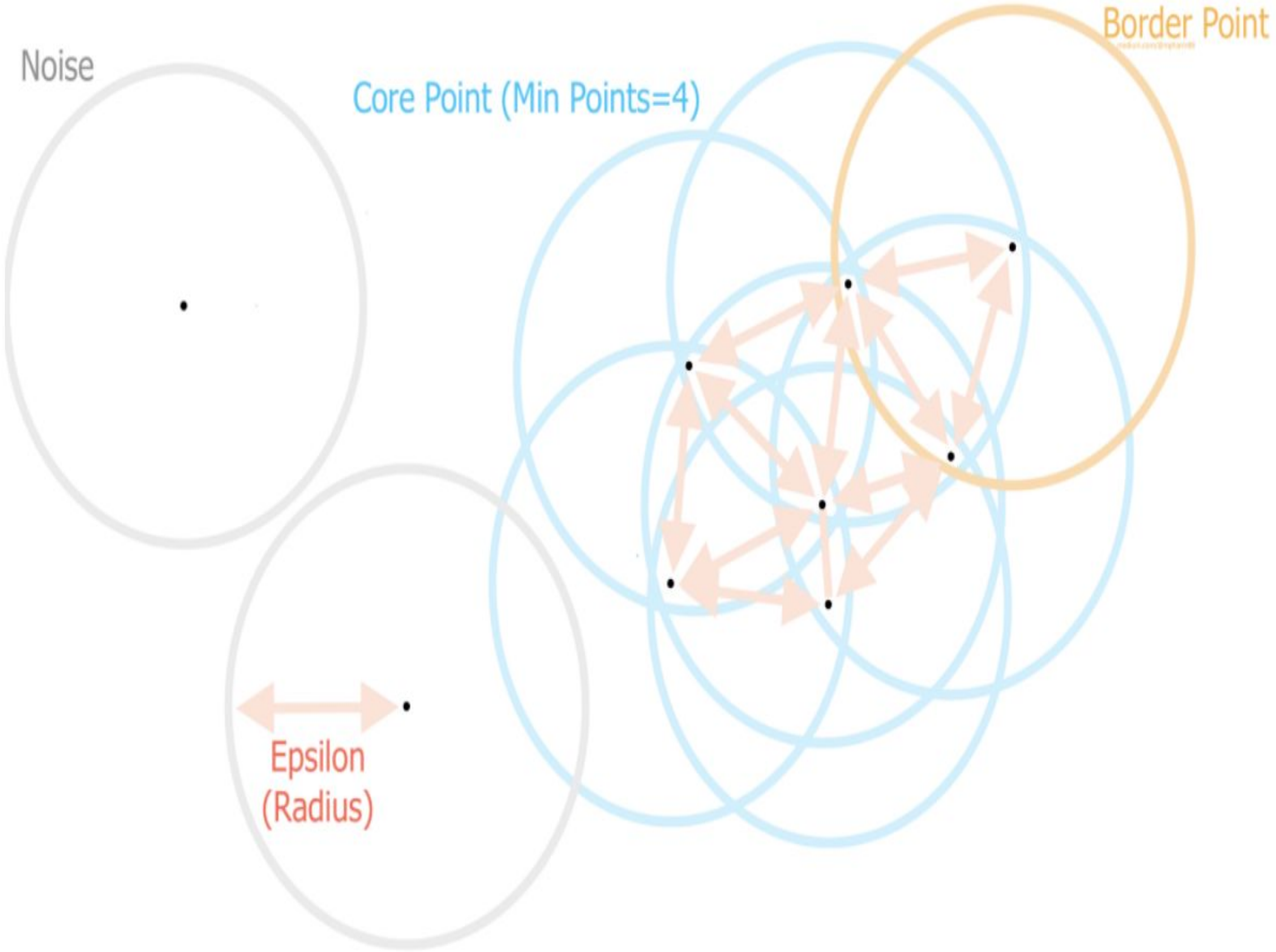
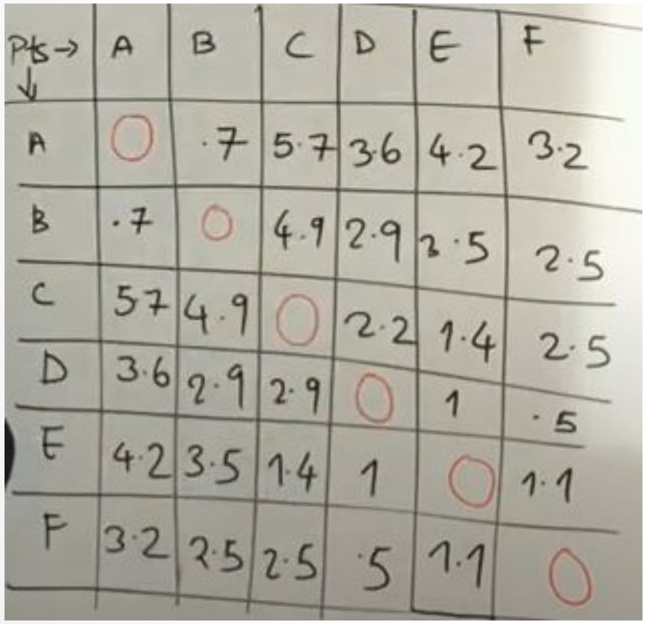
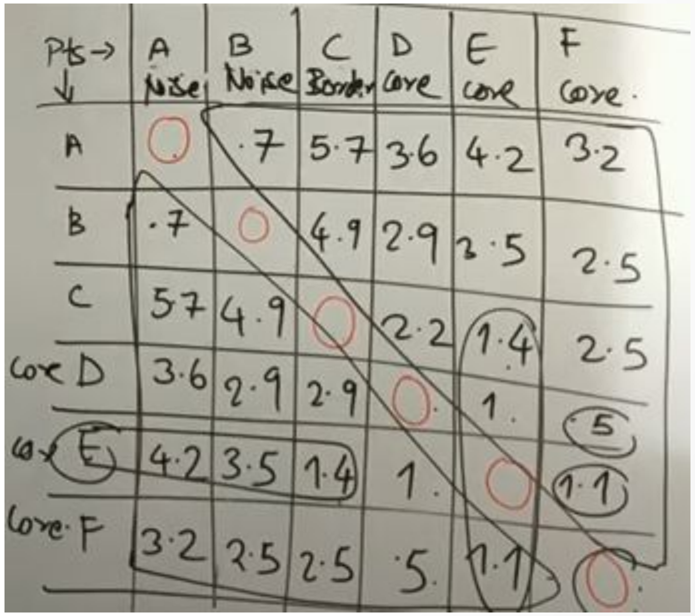


Image by Mikio Harman

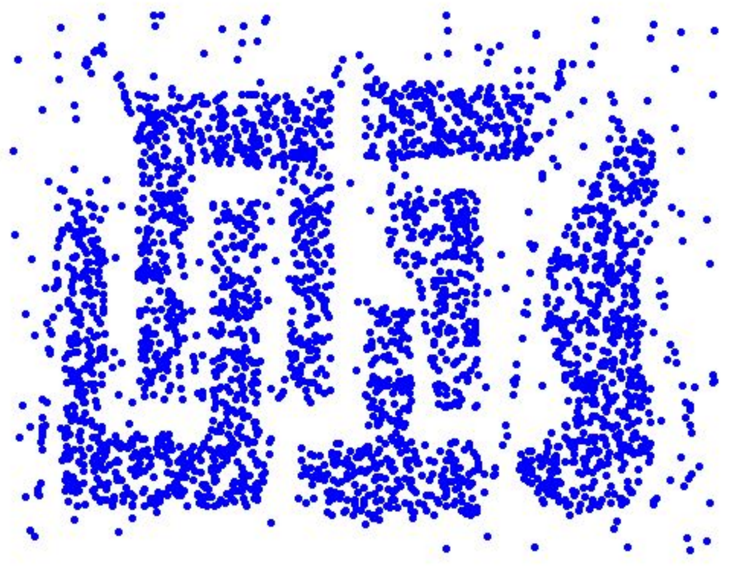


Pts→ ↓	A	B	C	D	E	F
A	0	·7	5·7	3·6	4·2	3·2
B	·7	0	4·9	2·9	3·5	2·5
C	5·7	4·9	0	2·2	1·4	2·5
D	3·6	2·9	2·9	0	1	·5
E	4·2	3·5	1·4	1	0	1·1
F	3·2	2·5	2·5	·5	1·1	0

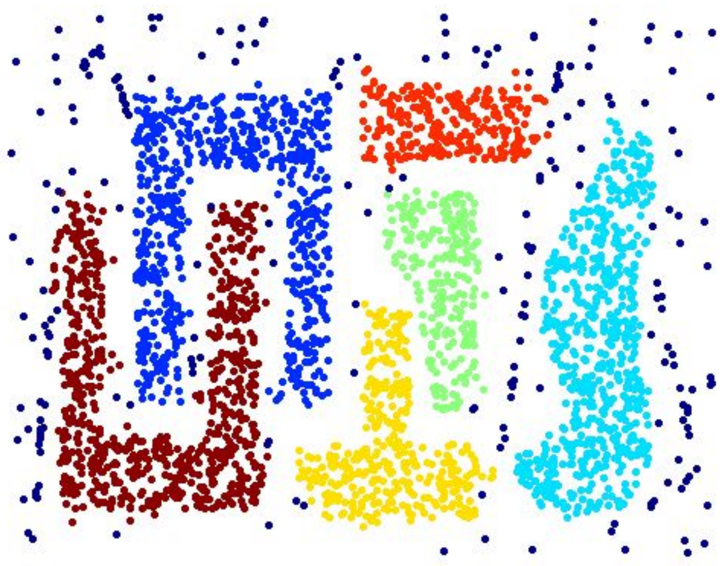


Pts→ ↓	A Noise	B Noise	C Border	D Core	E core	F Core.
A	0	·7	5·7	3·6	4·2	3·2
B	·7	0	4·9	2·9	3·5	2·5
C	5·7	4·9	0	2·2	1·4	2·5
Core D	3·6	2·9	2·9	0	1	·5
Core E	4·2	3·5	1·4	1	0	1·1
Core F	3·2	2·5	2·5	·5	1·1	0

When DBSCAN Works Well



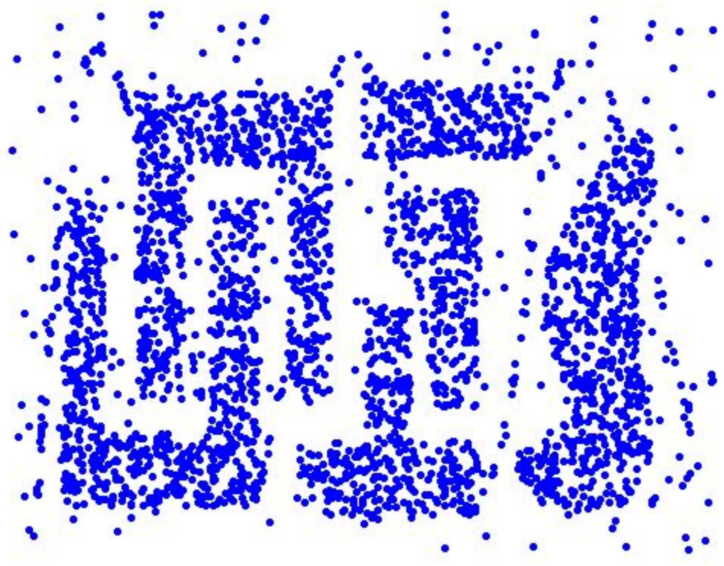
Original Points



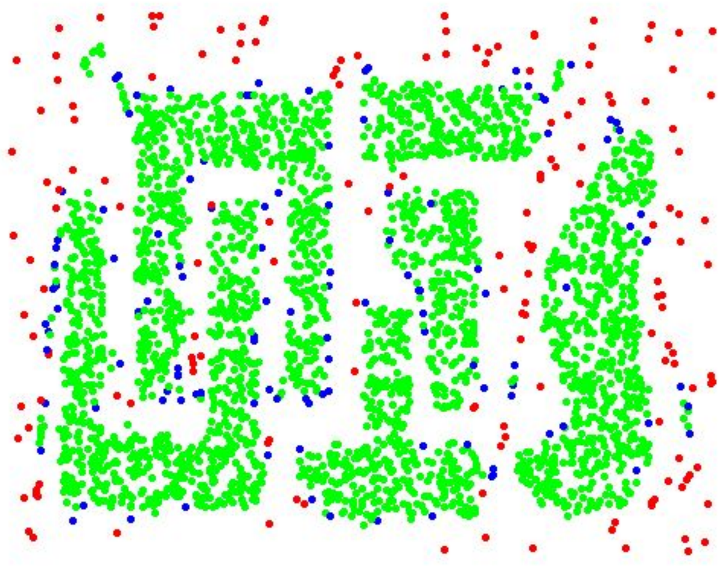
Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

DBSCAN: Core, Border and Noise Points



Original Points

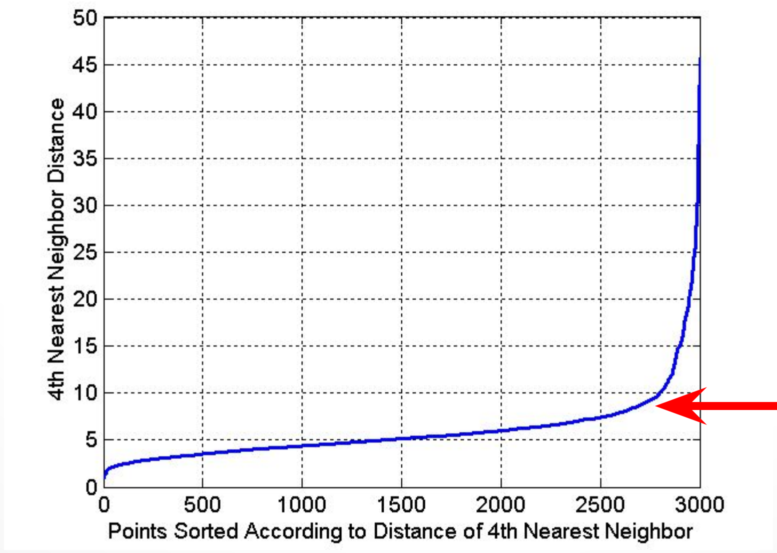


Point types: core, border and noise

Eps = 10, MinPts = 4

DBSCAN: Determining Eps and MinPts

- Idea is that for points in a cluster, their k^th nearest neighbors are at roughly the same distance
- Noise points have the k^th nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^th nearest neighbor
- Find the distance d where there is a "knee" in the curve
    - Eps = d, MinPts = k



Eps ~ 7-10
MinPts = 4

- DBSCAN, or Density-Based Spatial Clustering of Applications with Noise, is an unsupervised machine learning algorithm. Unsupervised machine learning algorithms are used to classify unlabeled data.
- DBSCAN is particularly well suited for problems which require:

1. Minimal domain knowledge to determine the input parameters (i.e. **K** in k-means and **Dmin** in hierarchical clustering)
2. Discovery of clusters with arbitrary shapes
3. Good efficiency on large databases